9 most useful functions for PySpark DataFrame

Pyspark

PySpark is a data analytics tool created by Apache Spark Community for using Python along with Spark. It allows us to work with RDD (Resilient Distributed Dataset) and DataFrames in Python. PySpark has numerous features that make it such an amazing framework and when it comes to deal with the huge amount of data PySpark provides us fast and Real-time processing, flexibility, in-memory computation, and various other features. It is a Python library to use Spark which combines the simplicity of Python language with the efficiency of Spark.

Pyspark DataFrame

A DataFrame is a distributed collection of data in rows under named columns. In simple terms, we can say that it is the same as a table in a Relational database or an Excel sheet with Column headers. DataFrames are mainly designed for processing a large-scale collection of structured or semi-structured data.

In this article, we’ll discuss functions of PySpark that are most useful and essential to perform efficient data analysis of structured data.

We are using Google Colab as the IDE for this data analysis.

We first need to install PySpark in Google Colab. After that, we will import the pyspark.sql module and create a SparkSession which will be an entry point of Spark SQL API.

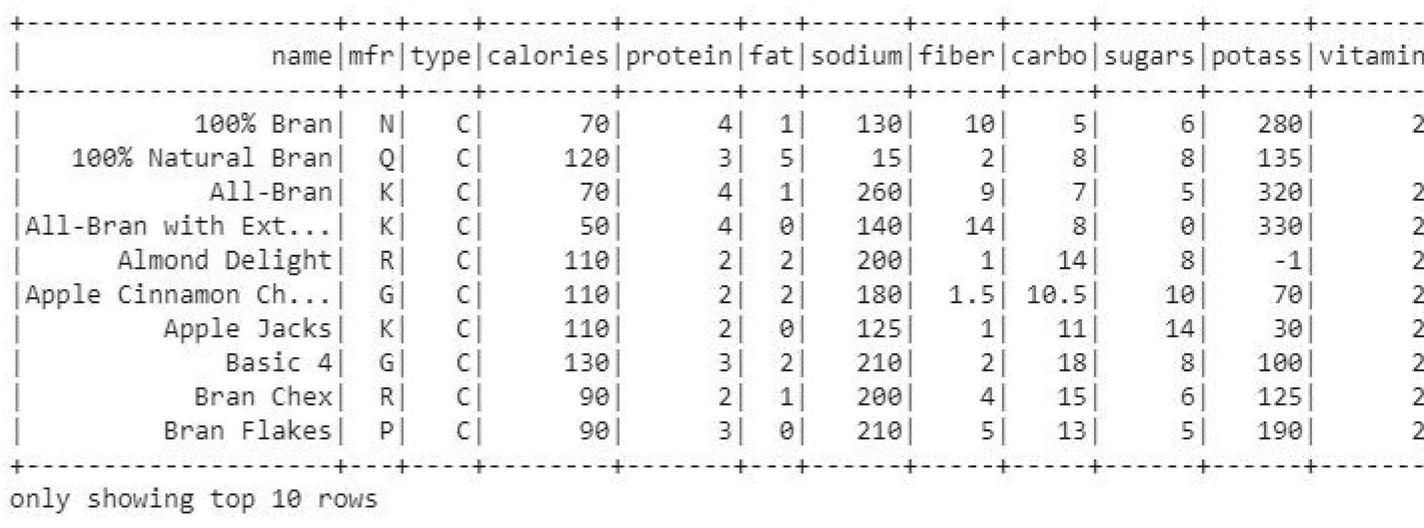
```
#installing pyspark
!pip install pyspark
```

```
#importing pyspark
import pyspark

#importing sparksessio
from pyspark.sql import SparkSession

#creating a sparksession object and providing appName
spark=SparkSession.builder.appName("pysparkdf").getOrCreate()
```

This SparkSession object will interact with the functions and methods of Spark SQL. Now, let’s create a Spark DataFrame by reading a CSV file. We will be using simple dataset i.e. Nutrition Data on 80 Cereal products available on Kaggle.

```
#creating a dataframe using spark object by reading csv file
df = spark.read.option("header", "true").csv("/content/cereal.csv")
```

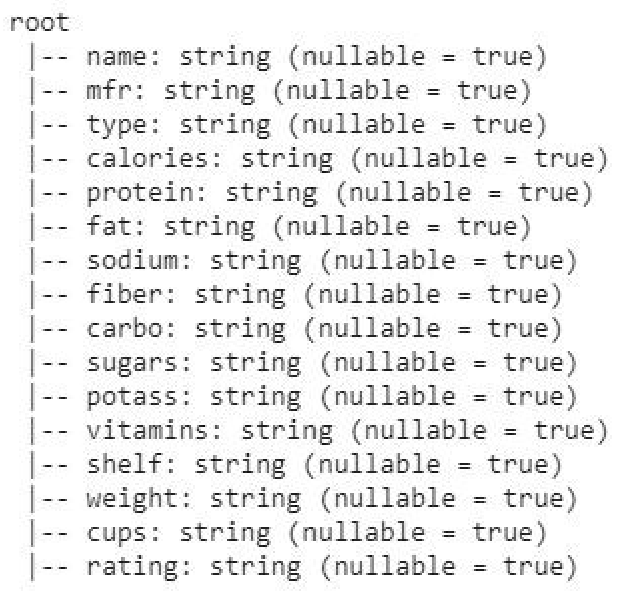
```
#show df created top 10 rows
df.show(10)
```



```
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
|                name|mfr|type|calories|protein|fat|sodium|fiber|carbo|sugars|potass|vitamin
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
|           100% Bran|  N|   C|      70|      4|  1|   130|   10|    5|     6|   280|      2
|   100% Natural Bran|  Q|   C|     120|      3|  5|    15|    2|    8|     8|   135|
|            All-Bran|  K|   C|      70|      4|  1|   260|    9|    7|     5|   320|      2
|All-Bran with Ext...|  K|   C|      50|      4|  0|   140|   14|    8|     0|   330|      2
|      Almond Delight|  R|   C|     110|      2|  2|   200|    1|   14|     8|    -1|      2
|Apple Cinnamon Ch...|  G|   C|     110|      2|  2|   180|  1.5| 10.5|    10|    70|      2
|         Apple Jacks|  K|   C|     110|      2|  0|   125|    1|   11|    14|    30|      2
|             Basic 4|  G|   C|     130|      3|  2|   210|    2|   18|     8|   100|      2
|           Bran Chex|  R|   C|      90|      2|  1|   200|    4|   15|     6|   125|      2
|         Bran Flakes|  P|   C|      90|      3|  0|   210|    5|   13|     5|   190|      2
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
only showing top 10 rows
```

This is the Dataframe we are using for Data analysis. Now, let’s print the schema of the DataFrame to know more about the dataset.

```
df.printSchema()
```



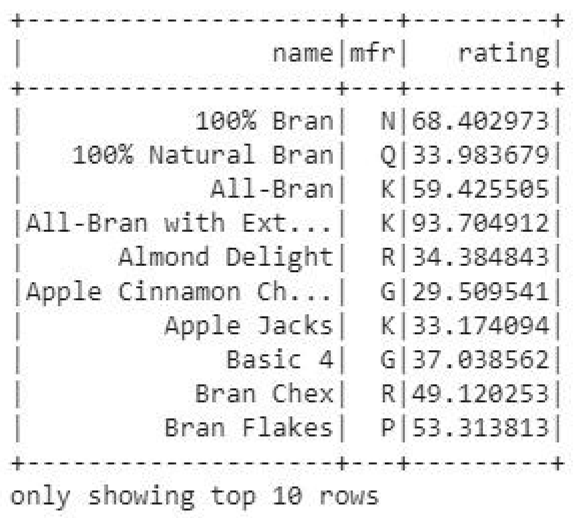
```
root
 |-- name: string (nullable = true)
 |-- mfr: string (nullable = true)
 |-- type: string (nullable = true)
 |-- calories: string (nullable = true)
 |-- protein: string (nullable = true)
 |-- fat: string (nullable = true)
 |-- sodium: string (nullable = true)
 |-- fiber: string (nullable = true)
 |-- carbo: string (nullable = true)
 |-- sugars: string (nullable = true)
 |-- potass: string (nullable = true)
 |-- vitamins: string (nullable = true)
 |-- shelf: string (nullable = true)
 |-- weight: string (nullable = true)
 |-- cups: string (nullable = true)
 |-- rating: string (nullable = true)
```

The DataFrame consists of 16 features or columns. Each column contains string-type values.

Let’s get started with the functions:

•**select():** The select function helps us to display a subset of selected columns from the entire dataframe we just need to pass the desired column names. Let’s print any three columns of the dataframe using select().

```
df.select('name', 'mfr', 'rating').show(10)
```



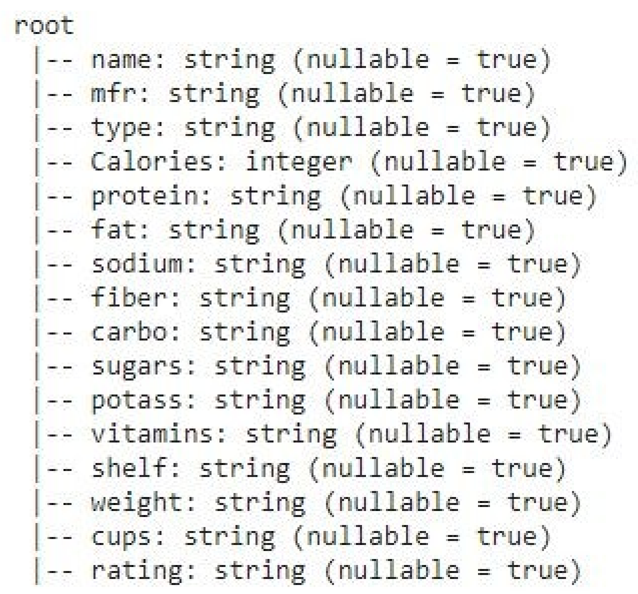
```
+--------------------+---+---------+
|                name|mfr|   rating|
+--------------------+---+---------+
|           100% Bran|  N|68.402973|
|   100% Natural Bran|  Q|33.983679|
|            All-Bran|  K|59.425505|
|All-Bran with Ext...|  K|93.704912|
|      Almond Delight|  R|34.384843|
|Apple Cinnamon Ch...|  G|29.509541|
|         Apple Jacks|  K|33.174094|
|             Basic 4|  G|37.038562|
|           Bran Chex|  R|49.120253|
|         Bran Flakes|  P|53.313813|
+--------------------+---+---------+
only showing top 10 rows
```

In the output, we got the subset of the dataframe with three columns name, mfr, rating.

•**withColumn():** The withColumn function is used to manipulate a column or to create a new column with the existing column. It is a transformation function, we can also change the datatype of any existing column.

In the DataFrame schema, we saw that all the columns are of string type. Let’s change the data type of calorie column to an integer.

```
df.withColumn("Calories",df['calories'].cast("Integer")).printSchema()
```



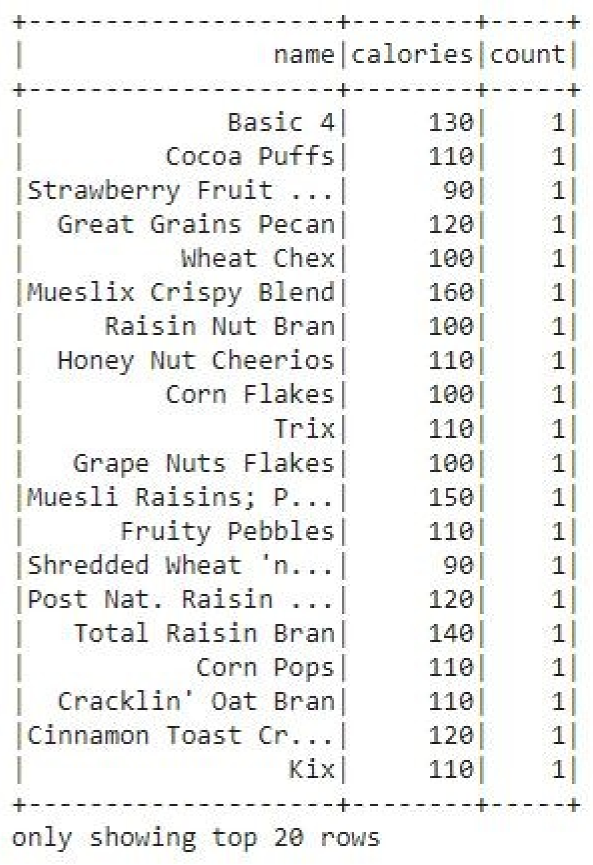
```
root
 |-- name: string (nullable = true)
 |-- mfr: string (nullable = true)
 |-- type: string (nullable = true)
 |-- Calories: integer (nullable = true)
 |-- protein: string (nullable = true)
 |-- fat: string (nullable = true)
 |-- sodium: string (nullable = true)
 |-- fiber: string (nullable = true)
 |-- carbo: string (nullable = true)
 |-- sugars: string (nullable = true)
 |-- potass: string (nullable = true)
 |-- vitamins: string (nullable = true)
 |-- shelf: string (nullable = true)
 |-- weight: string (nullable = true)
 |-- cups: string (nullable = true)
 |-- rating: string (nullable = true)
```

In the schema, we can see that the Datatype of calories column is changed to the integer type.

•**groupBy():** The groupBy function is used to collect the data into groups on DataFrame and allows us to perform aggregate functions on the grouped data. This is a very common data analysis operation similar to groupBy clause in SQL.

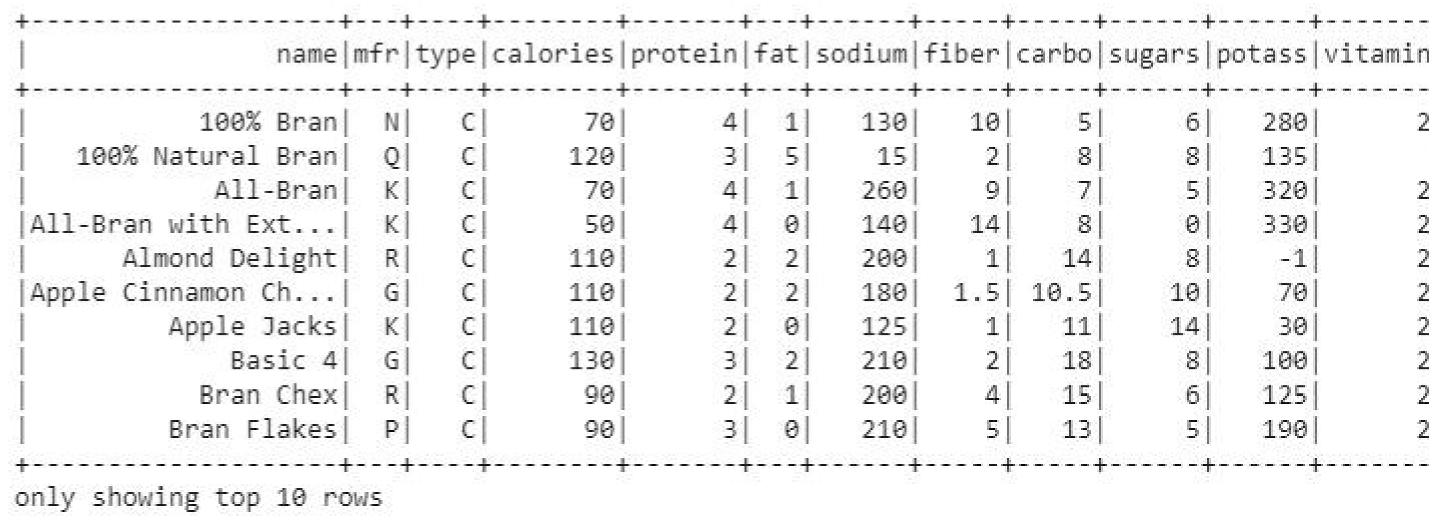
Let’s find out the count of each cereal present in the dataset.

```
df.groupBy("name", "calories").count().show()
```



```
+--------------------+--------+-----+
|                name|calories|count|
+--------------------+--------+-----+
|             Basic 4|     130|    1|
|         Cocoa Puffs|     110|    1|
|Strawberry Fruit ...|      90|    1|
|  Great Grains Pecan|     120|    1|
|          Wheat Chex|     100|    1|
|Mueslix Crispy Blend|     160|    1|
|     Raisin Nut Bran|     100|    1|
|  Honey Nut Cheerios|     110|    1|
|         Corn Flakes|     100|    1|
|                Trix|     110|    1|
|   Grape Nuts Flakes|     100|    1|
|Muesli Raisins; P...|     150|    1|
|      Fruity Pebbles|     110|    1|
|Shredded Wheat 'n...|      90|    1|
|Post Nat. Raisin ...|     120|    1|
|   Total Raisin Bran|     140|    1|
|           Corn Pops|     110|    1|
|  Cracklin' Oat Bran|     110|    1|
|Cinnamon Toast Cr...|     120|    1|
|                 Kix|     110|    1|
+--------------------+--------+-----+
only showing top 20 rows
```

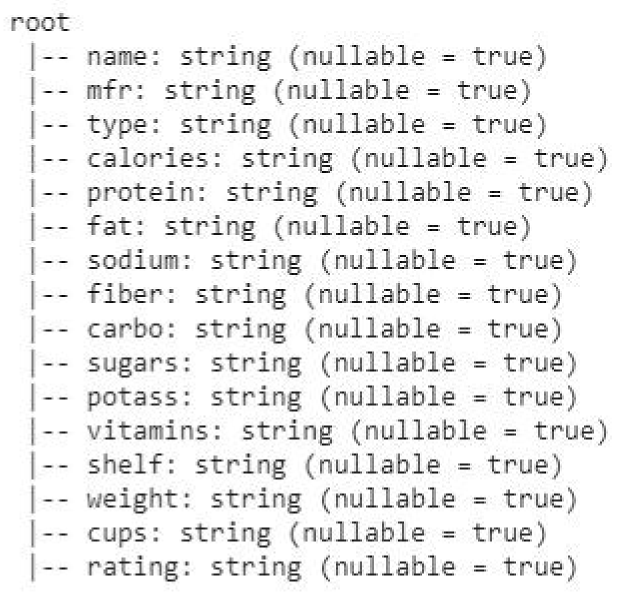
•**orderBy():** The orderBy function is used to sort the entire dataframe based on the particular column of the dataframe. It sorts the rows of the dataframe according to column values. By default, it sorts in ascending order.



```
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
|                name|mfr|type|calories|protein|fat|sodium|fiber|carbo|sugars|potass|vitamin
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
|           100% Bran|  N|   C|      70|      4|  1|   130|   10|    5|     6|   280|      2
|   100% Natural Bran|  Q|   C|     120|      3|  5|    15|    2|    8|     8|   135|
|            All-Bran|  K|   C|      70|      4|  1|   260|    9|    7|     5|   320|      2
|All-Bran with Ext...|  K|   C|      50|      4|  0|   140|   14|    8|     0|   330|      2
|      Almond Delight|  R|   C|     110|      2|  2|   200|    1|   14|     8|    -1|      2
|Apple Cinnamon Ch...|  G|   C|     110|      2|  2|   180|  1.5| 10.5|    10|    70|      2
|         Apple Jacks|  K|   C|     110|      2|  0|   125|    1|   11|    14|    30|      2
|             Basic 4|  G|   C|     130|      3|  2|   210|    2|   18|     8|   100|      2
|           Bran Chex|  R|   C|      90|      2|  1|   200|    4|   15|     6|   125|      2
|         Bran Flakes|  P|   C|      90|      3|  0|   210|    5|   13|     5|   190|      2
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
only showing top 10 rows
```

This is the Dataframe we are using for Data analysis. Now, let's print the schema of the DataFrame to know more about the dataset.

```
df.printSchema()
```



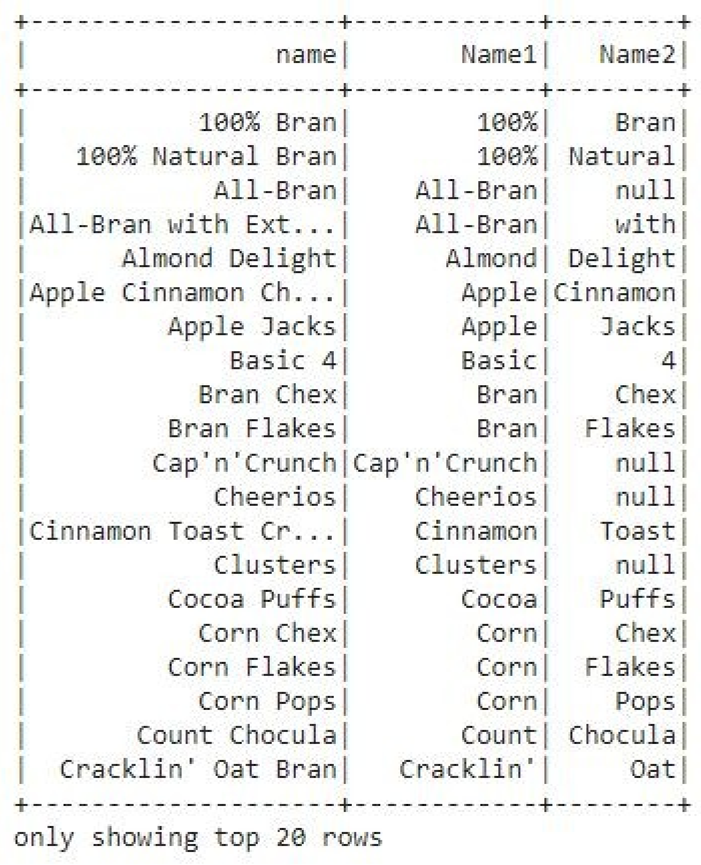
```
root
 |-- name: string (nullable = true)
 |-- mfr: string (nullable = true)
 |-- type: string (nullable = true)
 |-- calories: string (nullable = true)
 |-- protein: string (nullable = true)
 |-- fat: string (nullable = true)
 |-- sodium: string (nullable = true)
 |-- fiber: string (nullable = true)
 |-- carbo: string (nullable = true)
 |-- sugars: string (nullable = true)
 |-- potass: string (nullable = true)
 |-- vitamins: string (nullable = true)
 |-- shelf: string (nullable = true)
 |-- weight: string (nullable = true)
 |-- cups: string (nullable = true)
 |-- rating: string (nullable = true)
```

The DataFrame consists of 16 features or columns. Each column contains string-type values.

•**split():** The split() is used to split a string column of the dataframe into multiple columns. This function is applied to the dataframe with the help of withColumn() and select().

The name column of the dataframe contains values in two string words. Let's split the name column into two columns from space between two strings.

```
fropm pyspark.sql.functions import split


df1 = df.withColumn('Name1', split(df['name'], " ").getItem(0))
        .withColumn('Name2', split(df['name'], " ").getItem(1))


df1.select("name", "Name1", "Name2").show()
```



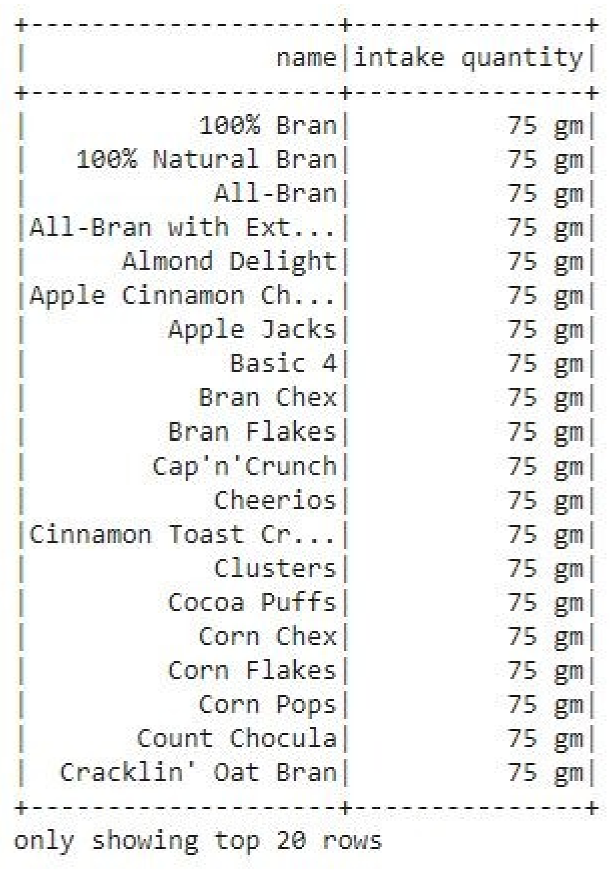
```
+--------------------+------------+--------+
|                name|       Name1|   Name2|
+--------------------+------------+--------+
|           100% Bran|        100%|    Bran|
|   100% Natural Bran|        100%| Natural|
|            All-Bran|    All-Bran|    null|
|All-Bran with Ext...|    All-Bran|    with|
|      Almond Delight|      Almond| Delight|
|Apple Cinnamon Ch...|       Apple|Cinnamon|
|         Apple Jacks|       Apple|   Jacks|
|             Basic 4|       Basic|       4|
|           Bran Chex|        Bran|    Chex|
|         Bran Flakes|        Bran|  Flakes|
|        Cap'n'Crunch|Cap'n'Crunch|    null|
|            Cheerios|    Cheerios|    null|
|Cinnamon Toast Cr...|    Cinnamon|   Toast|
|            Clusters|    Clusters|    null|
|         Cocoa Puffs|       Cocoa|   Puffs|
|           Corn Chex|        Corn|    Chex|
|         Corn Flakes|        Corn|  Flakes|
|           Corn Pops|        Corn|    Pops|
|       Count Chocula|       Count| Chocula|
|  Cracklin' Oat Bran|   Cracklin'|     Oat|
+--------------------+------------+--------+
only showing top 20 rows
```

In this output, we can see that the name column is split into columns.

- **lit():** The lit function is used to add a new column to the dataframe that contains literals or some constant value.

  Let's add a column "intake quantity" which contains a constant value for each of the cereals along with the respective cereal name.

```
from pyspark.sql.functions import lit

df2 = df.select(col("name"),lit("75 gm").alias("intake quantity"))
df2.show()
```



```
+--------------------+---------------+
|                name|intake quantity|
+--------------------+---------------+
|           100% Bran|          75 gm|
|   100% Natural Bran|          75 gm|
|            All-Bran|          75 gm|
|All-Bran with Ext...|          75 gm|
|      Almond Delight|          75 gm|
|Apple Cinnamon Ch...|          75 gm|
|         Apple Jacks|          75 gm|
|             Basic 4|          75 gm|
|           Bran Chex|          75 gm|
|         Bran Flakes|          75 gm|
|        Cap'n'Crunch|          75 gm|
|            Cheerios|          75 gm|
|Cinnamon Toast Cr...|          75 gm|
|            Clusters|          75 gm|
|         Cocoa Puffs|          75 gm|
|           Corn Chex|          75 gm|
|         Corn Flakes|          75 gm|
|           Corn Pops|          75 gm|
|       Count Chocula|          75 gm|
|  Cracklin' Oat Bran|          75 gm|
+--------------------+---------------+
only showing top 20 rows
```

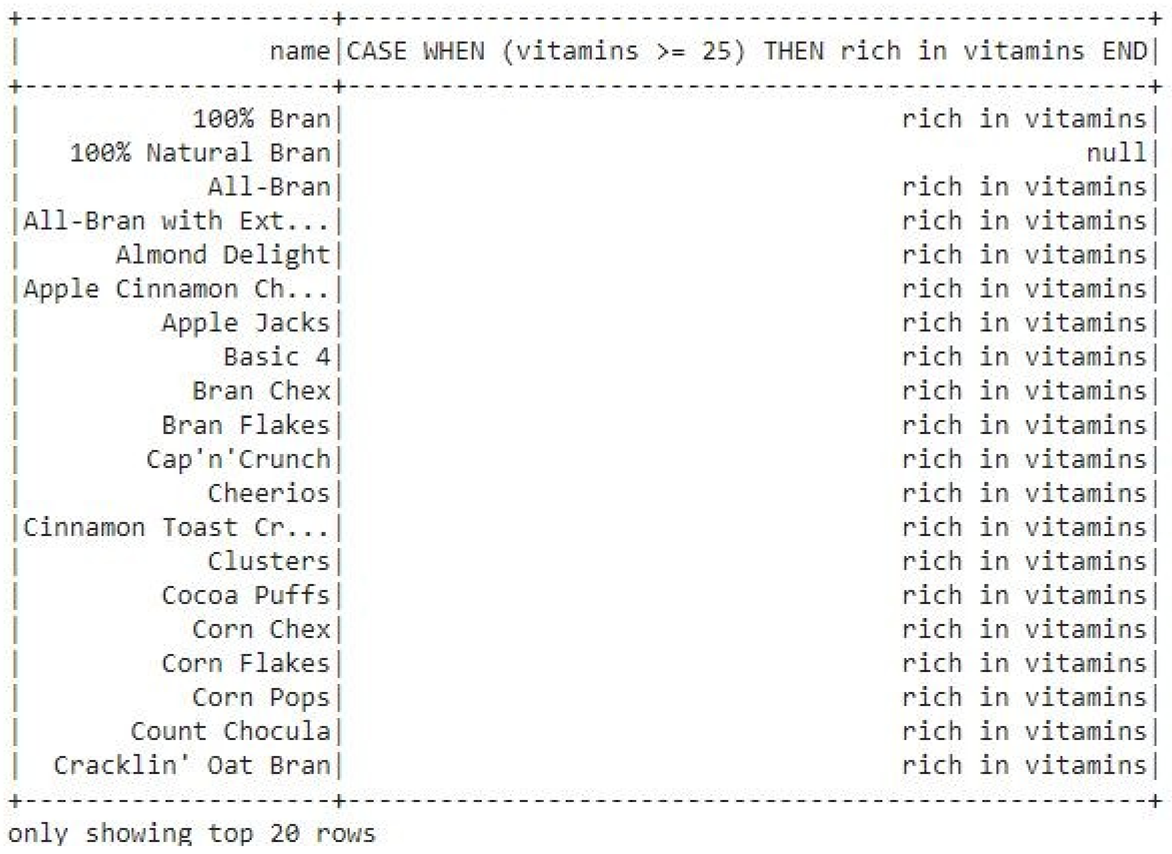
In the output, we can see that a new column is created “intak quantity” that contains the in-take a quantity of each cereal.

when(): The when the function is used to display the output based on the particular condition. It evaluates the condition provided and then returns the values accordingly. It is a SQL function that supports PySpark to check multiple conditions in a sequence and return the value. This function similarly works as if-then-else and switch statements.

Let’s see the cereals that are rich in vitamins.

```
from pyspark.sql.functions import when
```

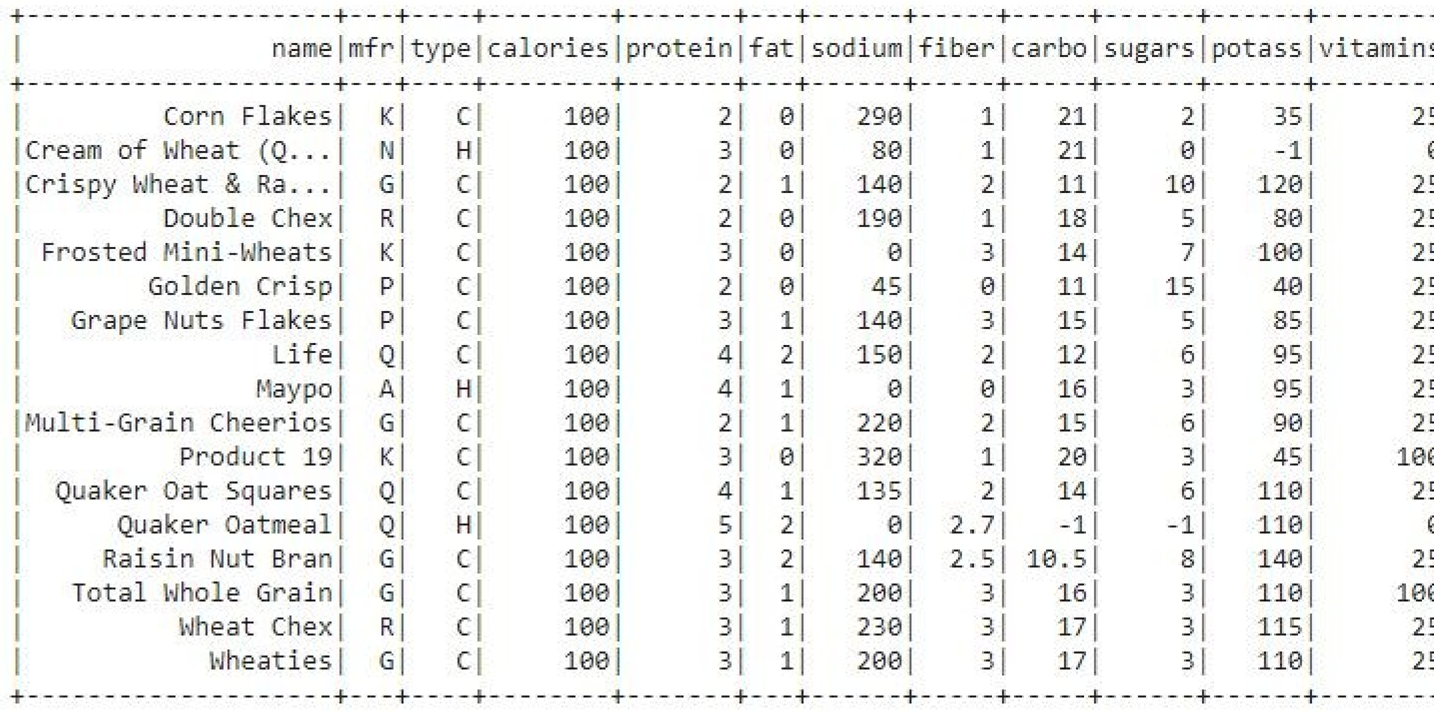
```
df.select("name", when(df.vitamins >= "25", "rich in vitamins")).show()
```


```
+--------------------+--------------------------------------------------+
|                name|CASE WHEN (vitamins >= 25) THEN rich in vitamins END|
+--------------------+--------------------------------------------------+
|           100% Bran|                                  rich in vitamins|
|   100% Natural Bran|                                              null|
|            All-Bran|                                  rich in vitamins|
|All-Bran with Ext...|                                  rich in vitamins|
|      Almond Delight|                                  rich in vitamins|
|Apple Cinnamon Ch...|                                  rich in vitamins|
|         Apple Jacks|                                  rich in vitamins|
|             Basic 4|                                  rich in vitamins|
|           Bran Chex|                                  rich in vitamins|
|         Bran Flakes|                                  rich in vitamins|
|        Cap'n'Crunch|                                  rich in vitamins|
|            Cheerios|                                  rich in vitamins|
|Cinnamon Toast Cr...|                                  rich in vitamins|
|            Clusters|                                  rich in vitamins|
|         Cocoa Puffs|                                  rich in vitamins|
|           Corn Chex|                                  rich in vitamins|
|         Corn Flakes|                                  rich in vitamins|
|           Corn Pops|                                  rich in vitamins|
|       Count Chocula|                                  rich in vitamins|
|  Cracklin' Oat Bran|                                  rich in vitamins|
+--------------------+--------------------------------------------------+
only showing top 20 rows
```

•**filter():** The filter function is used to filter data in rows based on the particular column values. For example, we can filter the cereals which have calories equal to 100.

```
from pyspark.sql.functions import filter
```

```
df.filter(df.calories == "100").show()
```



```
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+--------
|                name|mfr|type|calories|protein|fat|sodium|fiber|carbo|sugars|potass|vitamins
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+--------
|         Corn Flakes|  K|   C|     100|      2|  0|   290|    1|   21|     2|    35|      25
|Cream of Wheat (Q...|  N|   H|     100|      3|  0|    80|    1|   21|     0|    -1|       0
|Crispy Wheat & Ra...|  G|   C|     100|      2|  1|   140|    2|   11|    10|   120|      25
|         Double Chex|  R|   C|     100|      2|  0|   190|    1|   18|     5|    80|      25
| Frosted Mini-Wheats|  K|   C|     100|      3|  0|     0|    3|   14|     7|   100|      25
|        Golden Crisp|  P|   C|     100|      2|  0|    45|    0|   11|    15|    40|      25
|   Grape Nuts Flakes|  P|   C|     100|      3|  1|   140|    3|   15|     5|    85|      25
|                Life|  Q|   C|     100|      4|  2|   150|    2|   12|     6|    95|      25
|               Maypo|  A|   H|     100|      4|  1|     0|    0|   16|     3|    95|      25
|Multi-Grain Cheerios|  G|   C|     100|      2|  1|   220|    2|   15|     6|    90|      25
|          Product 19|  K|   C|     100|      3|  0|   320|    1|   20|     3|    45|     100
|  Quaker Oat Squares|  Q|   C|     100|      4|  1|   135|    2|   14|     6|   110|      25
|      Quaker Oatmeal|  Q|   H|     100|      5|  2|     0|  2.7|   -1|    -1|   110|       0
|     Raisin Nut Bran|  G|   C|     100|      3|  2|   140|  2.5| 10.5|     8|   140|      25
|   Total Whole Grain|  G|   C|     100|      3|  1|   200|    3|   16|     3|   110|     100
|          Wheat Chex|  R|   C|     100|      3|  1|   230|    3|   17|     3|   115|      25
|            Wheaties|  G|   C|     100|      3|  1|   200|    3|   17|     3|   110|      25
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+--------
```

In this output, we can see that the data is filtered according to the cereals which have

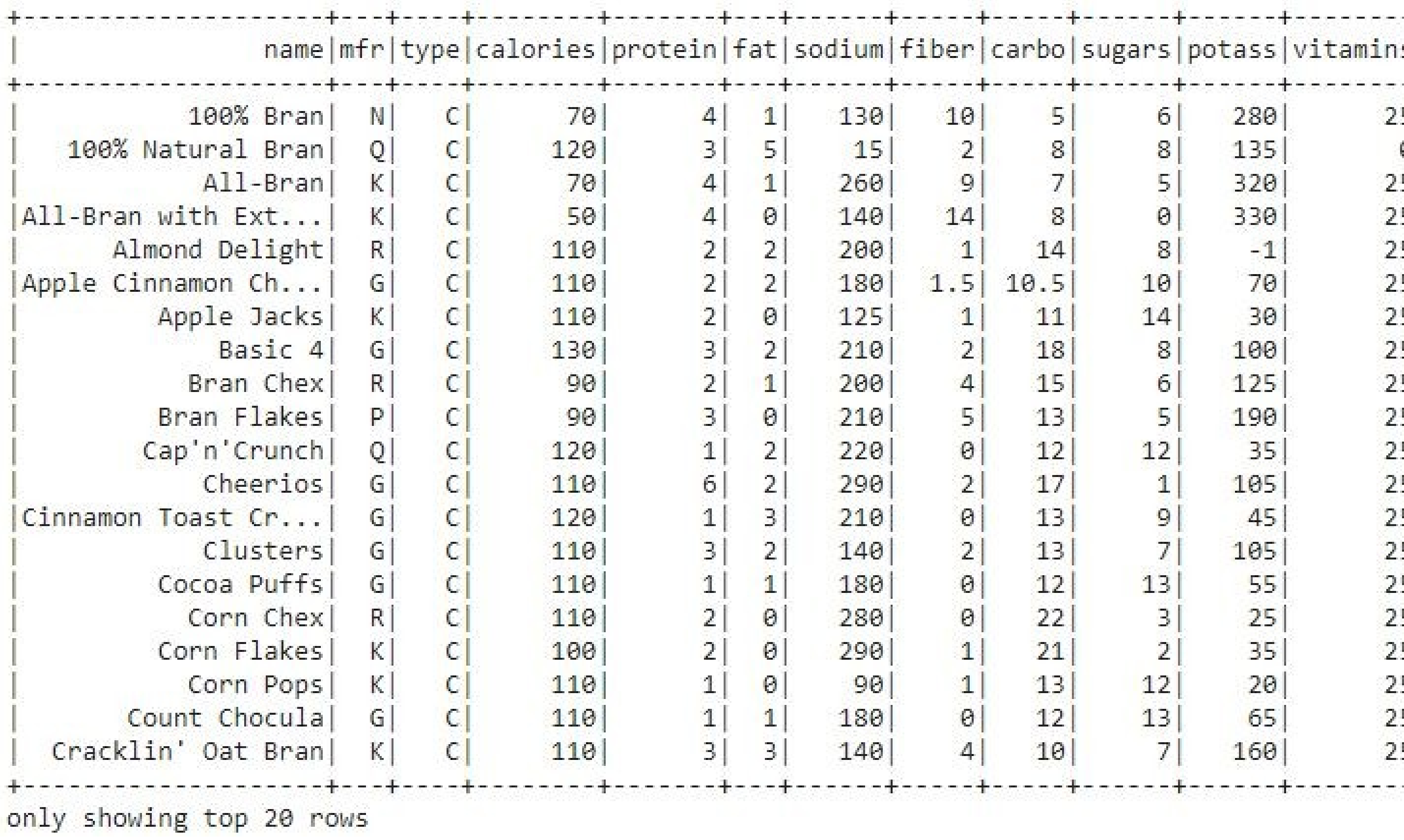
100 calories.

isNull()/isNotNull(): These two functions are used to find out if there is any null value present in the DataFrame. It is the most essential function for data processing. It is the major tool used for data cleaning.

Let's find out is there any null value present in the dataset.

```
#isNotNull()
```

```
from pyspark.sql.functions import *
#filter data by null values
df.filter(df.name.isNotNull()).show()
```

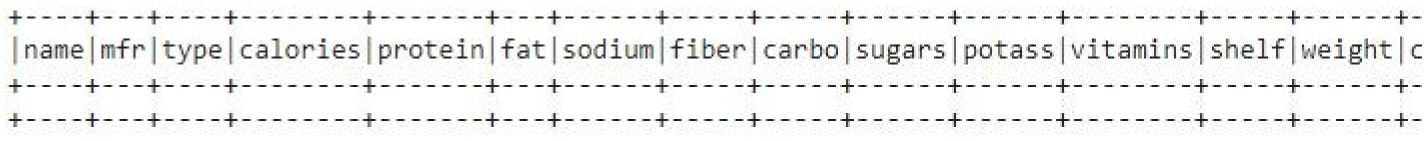


```
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
|                name|mfr|type|calories|protein|fat|sodium|fiber|carbo|sugars|potass|vitamin
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
|           100% Bran|  N|   C|      70|      4|  1|   130|   10|    5|     6|   280|       2
|   100% Natural Bran|  Q|   C|     120|      3|  5|    15|    2|    8|     8|   135|       
|            All-Bran|  K|   C|      70|      4|  1|   260|    9|    7|     5|   320|       2
|All-Bran with Ext...|  K|   C|      50|      4|  0|   140|   14|    8|     0|   330|       2
|      Almond Delight|  R|   C|     110|      2|  2|   200|    1|   14|     8|    -1|       2
|Apple Cinnamon Ch...|  G|   C|     110|      2|  2|   180|  1.5| 10.5|    10|    70|       2
|         Apple Jacks|  K|   C|     110|      2|  0|   125|    1|   11|    14|    30|       2
|             Basic 4|  G|   C|     130|      3|  2|   210|    2|   18|     8|   100|       2
|           Bran Chex|  R|   C|      90|      2|  1|   200|    4|   15|     6|   125|       2
|         Bran Flakes|  P|   C|      90|      3|  0|   210|    5|   13|     5|   190|       2
|        Cap'n'Crunch|  Q|   C|     120|      1|  2|   220|    0|   12|    12|    35|       2
|            Cheerios|  G|   C|     110|      6|  2|   290|    2|   17|     1|   105|       2
|Cinnamon Toast Cr...|  G|   C|     120|      1|  3|   210|    0|   13|     9|    45|       2
|            Clusters|  G|   C|     110|      3|  2|   140|    2|   13|     7|   105|       2
|         Cocoa Puffs|  G|   C|     110|      1|  1|   180|    0|   12|    13|    55|       2
|           Corn Chex|  R|   C|     110|      2|  0|   280|    0|   22|     3|    25|       2
|         Corn Flakes|  K|   C|     100|      2|  0|   290|    1|   21|     2|    35|       2
|           Corn Pops|  K|   C|     110|      1|  0|    90|    1|   13|    12|    20|       2
|       Count Chocula|  G|   C|     110|      1|  1|   180|    0|   12|    13|    65|       2
|  Cracklin' Oat Bran|  K|   C|     110|      3|  3|   140|    4|   10|     7|   160|       2
+--------------------+---+----+--------+-------+---+------+-----+-----+------+------+-------
only showing top 20 rows
```

There are no null values present in this dataset. Hence, the entire dataframe is displayed.

isNull():

```
df.filter(df.name.isNull()).show()
```



```
+----+---+----+--------+-------+---+------+-----+-----+------+------+--------+-----+------+-
|name|mfr|type|calories|protein|fat|sodium|fiber|carbo|sugars|potass|vitamins|shelf|weight|c
+----+---+----+--------+-------+---+------+-----+-----+------+------+--------+-----+------+-
+----+---+----+--------+-------+---+------+-----+-----+------+------+--------+-----+------+-
```

Again, there are no null values. Therefore, an empty dataframe is displayed.

In this blog, we have discussed the 9 most useful functions for efficient data processing. These PySpark functions are the combination of both the languages Python and SQL.