PySpark Filter() Function

PySpark `filter()` function is used to filter the rows from RDD/DataFrame based on the given condition or SQL expression, we can also use `where()` clause instead of the filter(). Both these functions operate the same. `filter()` function returns a new DataFrame or RDD with only the rows that meet the condition specified.

Before we start with examples, first let's create a DataFrame.

```
from pyspark.sql.types import StructType,StructField
from pyspark.sql.types import StringType, IntegerType, ArrayType
data = [
    (("James","","Smith"),["Java","Scala","C++"],"OH","M"),
    (("Anna","Rose",""),["Spark","Java","C++"],"NY","F"),
    (("Julia","","Williams"),["CSharp","VB"],"OH","F"),
    (("Maria","Anne","Jones"),["CSharp","VB"],"NY","M"),
    (("Jen","Mary","Brown"),["CSharp","VB"],"NY","M"),
    (("Mike","Mary","Williams"),["Python","VB"],"OH","M")
 ]

schema = StructType([
     StructField('name', StructType([
        StructField('firstname', StringType(), True),
        StructField('middlename', StringType(), True),
         StructField('lastname', StringType(), True)
     ])),
     StructField('languages', ArrayType(StringType()), True),
     StructField('state', StringType(), True),
     StructField('gender', StringType(), True)
 ])

df = spark.createDataFrame(data = data, schema = schema)
df.printSchema()
df.show(truncate=False)
```

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- languages: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- state: string (nullable = true)
 |-- gender: string (nullable = true)
```

```
+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Anna, Rose, }        |[Spark, Java, C++]|NY   |F     |
|{Julia, , Williams}   |[CSharp, VB]      |OH   |F     |
|{Maria, Anne, Jones}  |[CSharp, VB]      |NY   |M     |
|{Jen, Mary, Brown}    |[CSharp, VB]      |NY   |M     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
+----------------------+------------------+-----+------+
```

DataFrame filter() with Column Condition

```
1  df.filter(df.state == "OH").show(truncate=False)
```

▸ (1) Spark Jobs

```
+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Julia, , Williams}   |[CSharp, VB]      |OH   |F     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
+----------------------+------------------+-----+------+
```

```
1  df.filter(df.state != "OH") \
2      .show(truncate=False)
```

▸ (1) Spark Jobs

```
+--------------------+------------------+-----+------+
|name                |languages         |state|gender|
+--------------------+------------------+-----+------+
|{Anna, Rose, }      |[Spark, Java, C++]|NY   |F     |
|{Maria, Anne, Jones}|[CSharp, VB]      |NY   |M     |
|{Jen, Mary, Brown}  |[CSharp, VB]      |NY   |M     |
+--------------------+------------------+-----+------+
```

```
1  df.filter(~(df.state == "OH")) \
2      .show(truncate=False)
```

▸ (1) Spark Jobs

```
+--------------------+------------------+-----+------+
|name                |languages         |state|gender|
+--------------------+------------------+-----+------+
|{Anna, Rose, }      |[Spark, Java, C++]|NY   |F     |
|{Maria, Anne, Jones}|[CSharp, VB]      |NY   |M     |
|{Jen, Mary, Brown}  |[CSharp, VB]      |NY   |M     |
+--------------------+------------------+-----+------+
```

```
1  from pyspark.sql.functions import col
2  df.filter(col("state") == "OH") \
3      .show(truncate=False)
```

▸ (1) Spark Jobs

```
+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Julia, , Williams}   |[CSharp, VB]      |OH   |F     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
+----------------------+------------------+-----+------+
```

DataFrame filter() with SQL Expression

```
1  df.filter("gender == 'M'").show()
```

▸ (1) Spark Jobs

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

```
1  df.filter("gender != 'M'").show()
```

▸ (1) Spark Jobs

```
+------------------+-----------------+-----+------+
|              name|        languages|state|gender|
+------------------+-----------------+-----+------+
|    {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Julia, , Williams}|      [CSharp, VB]|   OH|     F|
+------------------+-----------------+-----+------+
```

```
1  df.filter("gender <> 'M'").show()
```

▸ (1) Spark Jobs

```
+------------------+-----------------+-----+------+
|              name|        languages|state|gender|
+------------------+-----------------+-----+------+
|    {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Julia, , Williams}|      [CSharp, VB]|   OH|     F|
+------------------+-----------------+-----+------+
```

PySpark Filter with Multiple Conditions

```
df.filter( (df.state  == "OH") & (df.gender  == "M") ) \
    .show(truncate=False)
```

▸ (1) Spark Jobs

```
+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
+----------------------+------------------+-----+------+
```

Filter Based on List Values

```
li=["OH","CA","DE"]
df.filter(df.state.isin(li)).show()
```

▸ (1) Spark Jobs

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
| {Julia, , Williams}|      [CSharp, VB]|   OH|     F|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

```
1  li=["OH","CA","DE"]
2  df.filter(~df.state.isin(li)).show()
```

▸ (1) Spark Jobs

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|      {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+-----------------+-----+------+
```

```
1  li=["OH","CA","DE"]
2  df.filter(df.state.isin(li)==False).show()
```

▸ (1) Spark Jobs

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|      {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+-----------------+-----+------+
```

Filter Based on Starts With, Ends With, Contains

```
1  # Using startswith
2  df.filter(df.state.startswith("N")).show()
```

▸ (1) Spark Jobs

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|      {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+------------------+-----+------+
```

```
1  #using endswith
2  df.filter(df.state.endswith("H")).show()
```

▸ (1) Spark Jobs

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
| {Julia, , Williams}|      [CSharp, VB]|   OH|     F|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

```
1  #contains
2  df.filter(df.state.contains("H")).show()
```

▸ (1) Spark Jobs

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
| {Julia, , Williams}|      [CSharp, VB]|   OH|     F|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+-----------------+-----+------+
```

PySpark Filter “like”

```
1  # like - SQL LIKE pattern
2  df.filter(df.name.firstname.like("%J%")).show()
```

▸ (1) Spark Jobs

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|   {James, , Smith}|[Java, Scala, C++]|   OH|     M|
|{Julia, , Williams}|      [CSharp, VB]|   OH|     F|
| {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+-----------------+-----+------+
```

Filter on an Array column

```
from pyspark.sql.functions import array_contains
df.filter(array_contains(df.languages,"Java")) \
    .show(truncate=False)
```

▸ (2) Spark Jobs

```
+----------------+------------------+-----+------+
|name            |languages         |state|gender|
+----------------+------------------+-----+------+
|{James, , Smith}|[Java, Scala, C++]|OH   |M     |
|{Anna, Rose, }  |[Spark, Java, C++]|NY   |F     |
+----------------+------------------+-----+------+
```

Filtering on Nested Struct columns

```
# Struct condition
df.filter(df.name.lastname == "Williams") \
    .show(truncate=False)
```

▸ (1) Spark Jobs

```
+----------------------+------------+-----+------+
|name                  |languages   |state|gender|
+----------------------+------------+-----+------+
|{Julia, , Williams}   |[CSharp, VB]|OH   |F     |
|{Mike, Mary, Williams}|[Python, VB]|OH   |M     |
+----------------------+------------+-----+------+
```