Apache Spark

PySpark – SQL Joins

5 months ago • by Gottumukkala Sravan Kumar

In Python, PySpark is a Spark module used to provide a similar kind of Processing like Spark using DataFrame.

In PySpark, SQL Joins are used to join two or more DataFrames based on the given condition. We just need to pass an SQL Query to perform different joins on the PySpark DataFrames. Spark.sql() is used to perform SQL Join in PySpark. Before that, we have to create a temporary view for the two PySpark DataFrames using createOrReplaceTempView() method. On this view, we can perform SQL Joins.

Syntax:

dataframe.createOrReplaceTempView("view_name")

Where:

1. DataFrame is the PySpark DataFrame.
2. view_name is the temporary name for the DataFrame.

There are different joins which are applied on the two PySpark DataFrames. Before going to know these types, let's create two PySpark DataFrames.

First DataFrame:

In the following example, we are going to create the PySpark DataFrame with 5 rows and 6 columns with student personal data and display using show() method:

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession
```

```
#create an app named linuxhint
spark_app = SparkSession.builder.appName(''linuxhint').getOrCreate()
# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi'','age':16,'height':3.79,'weight':34,'address':'hyd'},
               {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# dataframe
df.show()
```

Output:

```
+-------+---+------+----------------+------+------+
|address|age|height|            name|rollno|weight|
+-------+---+------+----------------+------+------+
| guntur| 23|  5.79|          sravan|   001|    67|
|    hyd| 16|  3.79|          ojaswi|   002|    34|
|  patna|  7|  2.79|gnanesh chowdary|   003|    17|
|    hyd|  9|  3.69|          rohith|   004|    28|
|    hyd| 37|  5.59|         sridevi|   005|    54|
+-------+---+------+----------------+------+------+
```

Second DataFrame

In the following example, we are going to create the PySpark DataFrame with 5 rows and 3 columns with student marks data and display using the show() method:

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# dataframe
df2.show()
```

Output:

```
+------+--------+--------+
|rollno|subject1|subject2|
+------+--------+--------+
|   001|      78|      98|
|   002|      83|      69|
|   005|      95|      90|
|   004|      76|     100|
|   007|      90|      91|
+------+--------+--------+
```

Observation:

From the two DataFrames, we have observed that rollno is the column that is common in both the DataFrames. So, we can apply the condition to this column. The rows — 001, 002, 004, and 005 — match in both DataFrames.

INNER JOIN

Inner Join results in the DataFrame by selecting only the matching rows from two DataFrames.
The keyword used is “inner”, which is the default join.

Syntax:

spark.sql(select * from v1 INNER JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to selecting all columns from both DataFrames.
2. v1 refers to the temporary view for the first DataFrame.
3. v2 refers to the temporary view for the second DataFrame.
4. Column_name is the column that exists common (common rows) in both DataFrames. (Based on this column, we will perform the join operation)

Example:

In this example, we are using the INNER JOIN keyword to join both DataFrames. So, the result will be 001, 002, 004, and 005 rows. Because they are common in both the DataFrames in the rollno column. Finally, we are applying the show() method to display the joined PySpark DataFrame.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
```

```
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('‘linuxhint').getOrCreate()

# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,''address':'hyd'},
                {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{''rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform inner join
spark.sql("select * from Student INNER JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

```
+-------+---+------+-------+------+------+------+--------+--------+
|address|age|height|   name|rollno|weight|rollno|subject1|subject2|
+-------+---+------+-------+------+------+------+--------+--------+
| guntur| 23|  5.79| sravan|   001|    67|   001|      78|      98|
|    hyd| 16|  3.79| ojaswi|   002|    34|   002|      83|      69|
|    hyd|  9|  3.69| rohith|   004|    28|   004|      76|     100|
|    hyd| 37|  5.59|sridevi|   005|    54|   005|      95|      90|
+-------+---+------+-------+------+------+------+--------+--------+
```

LEFT JOIN

Left Join results in the DataFrame by selecting all rows from the first DataFrame and only matching rows from the second DataFrame with respect to the rows in the first DataFrame.

Syntax:

spark.sql(select * from v1 LEFT JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to select all columns from both the DataFrames.
2. v1 refers to the temporary view for the first DataFrame.
3. v2 refers to the temporary view for the second DataFrame.
4. column_name is the column that exists common (common rows) in both the DataFrames. (Based on this column, we will perform the join operation)

Example:

In this example, we are using the LEFT JOIN keyword to join both DataFrames. So, the result will be 001,002, 003, 004, and 005 rows from the first DataFrame and 001, 002, 004, and 005 rows from the second DataFrame. Row 007 in the second DataFrame is not matched with any of the rows in the first DataFrame. So, null will be added in this row. Finally, we are applying the show() method to display the joined PySpark DataFrame.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()

# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,'address':'hyd'},
               {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform left join
spark.sql("select * from Student LEFT JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

```
+-------+---+------+---------------+------+------+------+--------+--------+
|address|age|height|           name|rollno|weight|rollno|subject1|subject2|
+-------+---+------+---------------+------+------+------+--------+--------+
| guntur| 23|  5.79|         sravan|   001|    67|   001|      78|      98|
|    hyd| 16|  3.79|         ojaswi|   002|    34|   002|      83|      69|
|  patna|  7|  2.79|gnanesh chowdary|   003|    17|  null|    null|    null|
|    hyd|  9|  3.69|         rohith|   004|    28|   004|      76|     100|
|    hyd| 37|  5.59|        sridevi|   005|    54|   005|      95|      90|
+-------+---+------+---------------+------+------+------+--------+--------+
```

LEFT OUTER JOIN

Left Outer Join results in the DataFrame by selecting all rows from the first DataFrame and only matching rows from the second DataFrame with respect to the rows in the first DataFrame. It is similar to the Left Join.

Syntax:

spark.sql(select * from v1 LEFT OUTER JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to select all columns from both the DataFrames.
2. v1 refers to the temporary view for the first DataFrame.
3. v2 refers to the temporary view for the second DataFrame.
4. column_name is the column that exists common (common rows) in both the DataFrames. (Based on this column, we will perform the join operation)

Example:

In this example, we are using the LEFT OUTER JOIN keyword to join both the DataFrames. So, the result will be 001, 002, 003, 004, and 005 rows from the first DataFrame and 001, 002, 004, and 005 rows from the second DataFrame. Row 007 in the second DataFrame is not matched with any row in the first DataFrame. So, null will be added in this row. Finally, we are applying the show() method to display the joined PySpark DataFrame.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()

# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},
```

```
 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,'address':'hyd'},
               {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform left outer join
spark.sql("select * from Student LEFT OUTER JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

```
+-------+---+------+----------------+------+------+------+--------+--------+
|address|age|height|            name|rollno|weight|rollno|subject1|subject2|
+-------+---+------+----------------+------+------+------+--------+--------+
| guntur| 23|  5.79|          sravan|   001|    67|   001|      78|      98|
|    hyd| 16|  3.79|          ojaswi|   002|    34|   002|      83|      69|
|  patna|  7|  2.79|gnanesh chowdary|   003|    17|  null|    null|    null|
|    hyd|  9|  3.69|          rohith|   004|    28|   004|      76|     100|
|    hyd| 37|  5.59|         sridevi|   005|    54|   005|      95|      90|
+-------+---+------+----------------+------+------+------+--------+--------+
```

RIGHT JOIN

Right Join results in the DataFrame by selecting all rows from the second DataFrame and only matching rows from the first DataFrame with respect to the rows in the second DataFrame. It places **null** values for the unmatched rows in the first DataFrame.

Syntax:

spark.sql(select * from v1 RIGHT JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to select all columns from both the DataFrames.
2. v1 refers to the temporary view for the first DataFrame.

3. v2 refers to the temporary view for the second DataFrame.
4. column_name is the column that exists common (common rows) in both the DataFrames. (Based on this column, we will perform the join operation)

Example:

In this example, we are using RIGHT JOIN keyword to join both DataFrames. So, the result will be 001, 002, 007, 004, and 005 rows from the second DataFrame and 001, 002, 004, and 005 rows from the first DataFrame. Row 003 in the first DataFrame is not matched with any of the row in the second DataFrame. So, null will be added in this row. Finally, we are applying the show() method to display the joined PySpark DataFrame.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()

# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,'address':'hyd'},
               {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform right join
spark.sql("select * from Student RIGHT JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

```
+-------+----+------+-------+------+------+------+--------+--------+
|address| age|height|   name|rollno|weight|rollno|subject1|subject2|
+-------+----+------+-------+------+------+------+--------+--------+
| guntur|  23|  5.79| sravan|   001|    67|   001|      78|      98|
|    hyd|  16|  3.79| ojaswi|   002|    34|   002|      83|      69|
|    hyd|   9|  3.69| rohith|   004|    28|   004|      76|     100|
|    hyd|  37|  5.59|sridevi|   005|    54|   005|      95|      90|
|   null|null|  null|   null|  null|  null|   007|      90|      91|
+-------+----+------+-------+------+------+------+--------+--------+
```

RIGHT OUTER JOIN

Right Outer Join results in the DataFrame by selecting all rows from the second DataFrame and only matching rows from the first DataFrame with respect to the rows in the second DataFrame. It places **null** values for the unmatched rows in the first DataFrame. It is similar to the Right Join.

Syntax:

spark.sql(select * from v1 RIGHT OUTER JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to select all columns from both the DataFrames.
2. v1 refers to the temporary view for the first DataFrame.
3. v2 refers to the temporary view for the second DataFrame.
4. column_name is the column that exists common (common rows) in both the DataFrames. (Based on this column, we will perform the join operation)

Example:

In this example, we are using RIGHT OUTER JOIN keyword to join both DataFrames. So, the result will be 001, 002, 007, 004, and 005 rows from the second DataFrame and 001, 002, 004, and 005 rows from the first DataFrame. Row 003 in first DataFrame is not matched with any of the row in the second DataFrame. So, null will be added in this row. Finally, we are applying the show() method to display the joined PySpark DataFrame.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()
```

```
# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,'address':'hyd'},
              {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform right outer join
spark.sql("select * from Student RIGHT OUTER JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

FULL JOIN

Full Join results in the DataFrame by selecting all rows from both the DataFrames. It places **null** values for the unmatched rows in both DataFrames across the rows.

Syntax:

spark.sql(select * from v1 FULL JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to select all columns from both the DataFrames.

2. v1 refers to the temporary view for the first DataFrame.
3. v2 refers to the temporary view for the second DataFrame.
4. column_name is the column that exists common (common rows) in both the DataFrames. (Based on this column, we will perform the join operation)

Example:

In the following example, we are using FULL JOIN keyword to join both DataFrames. So, the result will be from both DataFrames.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()

# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,'address':'hyd'},
               {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},

 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform full join
spark.sql("select * from Student FULL JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

FULL OUTER JOIN

Full Outer Join results in the DataFrame by selecting all rows from the DataFrames. It places **null** values for the unmatched rows in both DataFrames across the rows.

Syntax:

spark.sql(select * from v1 FULL OUTER JOIN v2 ON v1.column_name == v2.column_name)

Where:

1. * refers to select all columns from both the DataFrames.
2. v1 refers to the temporary view for the first DataFrame.
3. v2 refers to the temporary view for the second DataFrame.
4. column_name is the column that exists common (common rows) in both the DataFrames. (Based on this column, we will perform the join operation)

Example:

In this example, we are using FULL OUTER JOIN keyword to join both DataFrames. So, the result will be from both DataFrames.

```
#import the pyspark module
import pyspark
#import SparkSession for creating a session
from pyspark.sql import SparkSession

#create an app named linuxhint
spark_app = SparkSession.builder.appName('linuxhint').getOrCreate()

# create student data with 5 rows and 6 attributes
students =
[{'rollno':'001','name':'sravan','age':23,'height':5.79,'weight':67,'address':'guntur'},

 {'rollno':'002','name':'ojaswi','age':16,'height':3.79,'weight':34,'address':'hyd'},
               {'rollno':'003','name':'gnanesh
chowdary','age':7,'height':2.79,'weight':17,'address':'patna'},

 {'rollno':'004','name':'rohith','age':9,'height':3.69,'weight':28,'address':'hyd'},
```

```
 {'rollno':'005','name':'sridevi','age':37,'height':5.59,'weight':54,'address':'hyd'}]

# create the dataframe
df = spark_app.createDataFrame( students)

# create student_marks data with 5 rows and 6 attributes
students_marks =[{'rollno':'001','subject1':78,'subject2':98},
                 {'rollno':'002','subject1':83,'subject2':69},
                 {'rollno':'005','subject1':95,'subject2':90},
                 {'rollno':'004','subject1':76,'subject2':100},
                 {'rollno':'007','subject1':90,'subject2':91}]

# create the dataframe
df2 = spark_app.createDataFrame( students_marks)

# create view for df with names as Student
df.createOrReplaceTempView("Student")

# create view for df2 with names as Marks
df2.createOrReplaceTempView("Marks")

# perform full outer join
spark.sql("select * from Student FULL OUTER JOIN  Marks ON  Student.rollno ==
Marks.rollno").show()
```

Output:

Conclusion

In this article, we discussed four types of SQL Joins performed on PySpark DataFrame. We have seen that the LEFT JOIN is similar to the LEFT OUTER JOIN; the RIGHT JOIN is similar to the RIGHT OUTER JOIN; the FULL JOIN is similar to the FULL OUTER JOIN. We hope you found this article helpful. Check out other Linux Hint articles for more tips and tutorials.

ABOUT THE AUTHOR

Gottumukkala Sravan Kumar

B tech-hon's in Information Technology; Known programming languages - Python, R , PHP MySQL; Published 500+ articles on computer science domain

View all posts

RELATED LINUX HINT POSTS

- PySpark radians() and degrees() Functions
- PySpark desc_nulls_first() and desc_nulls_last() Functions
- PySpark – Row_Number() Function
- PySpark – Variance_Stddev() Function
- PySpark – Min() Function
- PySpark – Lag() Function
- PySpark – Greatest() Function

Linux Hint LLC, editor@linuxhint.com
1309 S Mary Ave Suite 210, Sunnyvale, CA 94087
Privacy Policy and Terms of Use