How can I query a database table using JDBC in PySpark?

Authored by

stats writer

June 24, 2024

JDBC (Java Database Connectivity) is a Java API that allows for the connection and interaction with a database. PySpark, a Python API for Apache Spark, also has the ability to query a database table using JDBC. This can be achieved by first establishing a JDBC connection to the database, then using the JDBC API methods to execute SQL queries on the database table. This process allows for the retrieval of data from the database table, which can then be used for further analysis and processing in PySpark. By utilizing JDBC in PySpark, users have the ability to seamlessly integrate data from external databases into their PySpark applications, enabling efficient and effective data processing.

To query a database table using JDBC in PySpark, you need to establish a connection to the database, specify the JDBC URL, and provide authentication credentials if required. PySpark's `read.jdbc()` method facilitates this process.

By using an option `dbtable` or `query` with jdbc() method you can do the SQL query on the database table into PySpark DataFrame.

Steps to query the database table using JDBC

1. PySpark Query JDBC Database Table

To query a database table using jdbc() method, you would need the following.

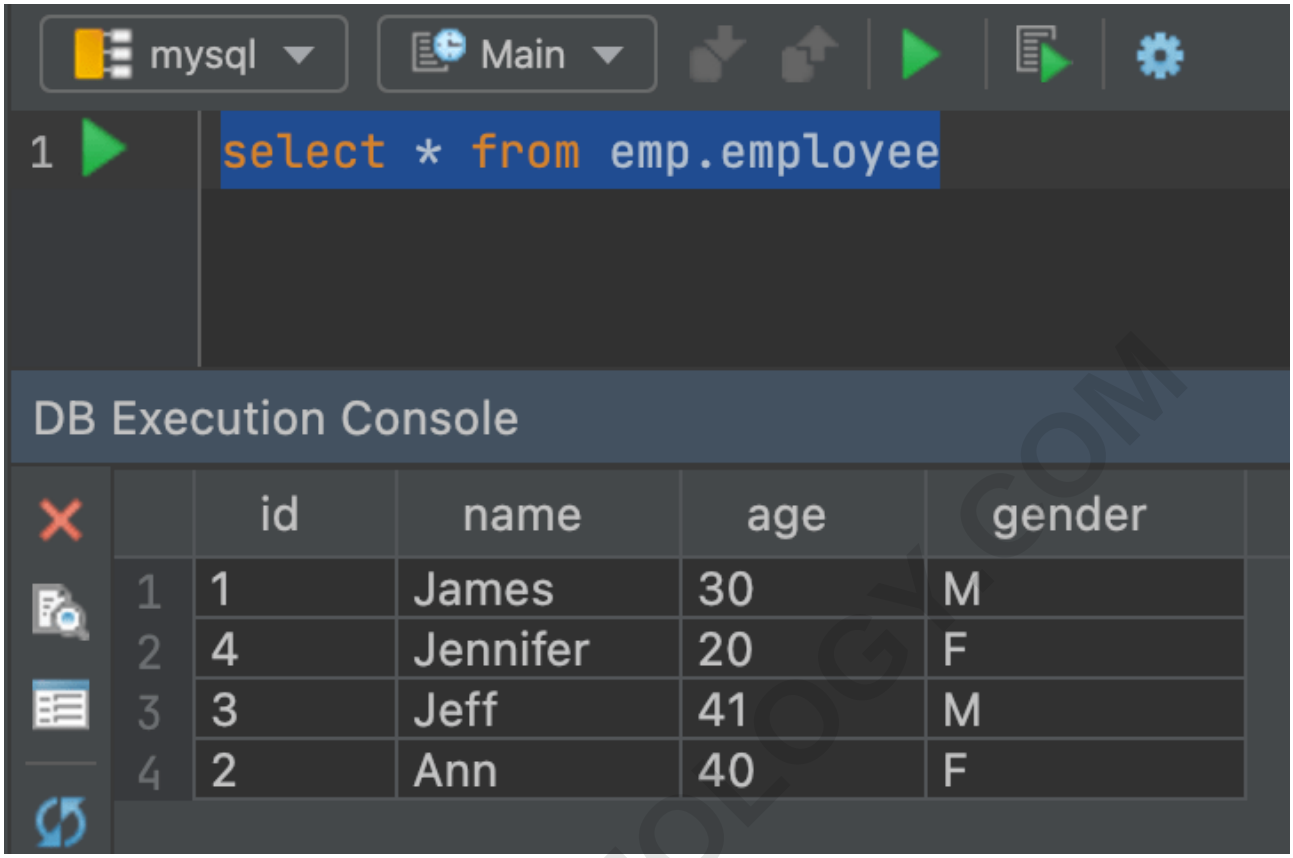
JDBC is a Java standard to connect to any database as long as you provide the right JDBC connector jar in the classpath and provide a JDBC driver using the JDBC API. PySpark also leverages the same JDBC standard when using jdbc() method.

The connector I am using in this article is `mysql-connector-java-<version>.jar` and the driver I am using `com.mysql.jdbc.Driver`

MySQL provides connectors for each server version hence, please choose the right version based pm server version you use. Download the `mysql-connector-java-8.0.13.jar` and keep it in your current directory.

2 PySpark Query JDBC Table Example

I have a MySQL database `emp` and table `employee` with column names `id`, `name`, `age` and `gender`.



I will use this JDBC table to run SQL queries and store the output in PySpark DataFrame. The below example extracts the complete table into DataFrame

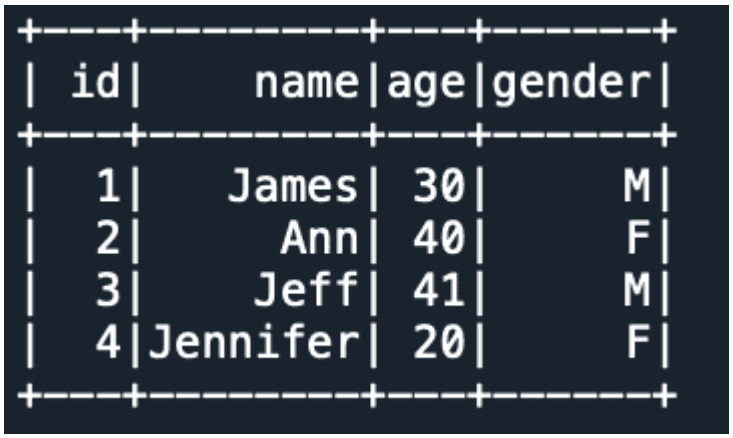
```
# Imports
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder
.appName('SparkByExamples.com')
.config("spark.jars", "mysql-connector-java-8.0.13.jar")
.getOrCreate()

# Query table using jdbc()
df = spark.read
.jdbc("jdbc:mysql://localhost:3306/emp", "employee",
properties={"user": "root", "password": "root", "driver":"com.mysql.cj.jdbc.Driver"})

# show DataFrame
df.show()
```

Yields below output. For more JDBC properties refer to https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html



```
+---+--------+---+------+
| id|    name|age|gender|
+---+--------+---+------+
|  1|   James| 30|     M|
|  2|     Ann| 40|     F|
|  3|    Jeff| 41|     M|
|  4|Jennifer| 20|     F|
+---+--------+---+------+
```

Alternatively, you can also use the `DataFrameReader.format("jdbc").load()` to query the table. When you use this, you need to provide the database details with the option() method.

```
# Query from MySQL Table
df = spark.read
.format("jdbc")
.option("url", "jdbc:mysql://localhost:3306/emp")
.option("driver", "com.mysql.cj.jdbc.Driver")
.option("dbtable", "employee")
.option("user", "root")
.option("password", "root")
.load()
```

3. SQL Query Specific Columns of JDBC Table

In the above example, it extracts the entire JDBC table into PySpark DataFrame. Sometimes you may be required to query specific columns with where condition. You can achieve this by using either `dbtable` or `query` options.

```
# Query from MySQL Table
df = spark.read
.format("jdbc")
.option("url", "jdbc:mysql://localhost:3306/emp")
.option("driver", "com.mysql.cj.jdbc.Driver")
.option("query", "select id,age from employee where gender='M'")
```

```
.option("user", "root")
.option("password", "root")
.load()
```

df.show()

Alternatively, you have the option to use the "`query`" parameter. It's important to note that you can only use either "`dbtable`" or "`query`" at a given time; you can't use both simultaneously. Additionally, if you opt for the "`query`" parameter, you won't be able to utilize the "`partitionColumn`" option.

```
# Using query
df = spark.read
.format("jdbc")
......
......
.option("query", "select id,age from employee where gender='M'")
.......
.......
.load()
```

4. Query JDBC Table Parallel

Use option `numPartitions` to query JDBC table in parallel.

```
# Using numPartitions
df = spark.read
.format("jdbc")
.option("query", "select id,age from employee where gender='M'")
.option("numPartitions",5)
.option("fetchsize", 20)
.......
.......
.load()
```

5. Complete Example

Following is the complete example of how to query a database table using jdbc() method in PySpark.

```
# Imports
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder
.appName('SparkByExamples.com')
.config("spark.jars", "mysql-connector-java-8.0.13.jar")
.getOrCreate()

# Query table using jdbc()
df = spark.read
.jdbc("jdbc:mysql://localhost:3306/emp", "employee",
properties={"user": "root", "password": "root", "driver":"com.mysql.cj.jdbc.Driver"})

# show DataFrame
df.show()

# Query from MySQL Table
df = spark.read
.format("jdbc")
.option("url", "jdbc:mysql://localhost:3306/emp")
.option("driver", "com.mysql.cj.jdbc.Driver")
.option("dbtable", "employee")
.option("user", "root")
.option("password", "root")
.load()

# Query from MySQL Table
df = spark.read
.format("jdbc")
.option("url", "jdbc:mysql://localhost:3306/emp")
.option("driver", "com.mysql.cj.jdbc.Driver")
.option("query", "select id,age from employee where gender='M'")
.option("user", "root")
.option("password", "root")
.load()

df.show()
```

Conclusion

In this article, you have learned how to SQL query a database table using jdbc() method in PySpark. Also, learned how to query the specific columns with where condition.

Related Articles

References