APACHE SPARK SQL

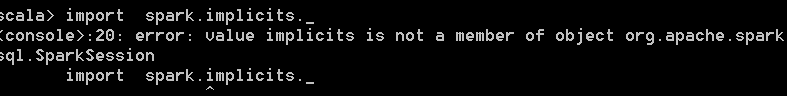
Spark SQL adalah modul Spark untuk pemrosesan data terstruktur. Berbeda dengan API Spark RDD dasar, antarmuka yang disediakan oleh Spark SQL memberikan Spark dengan lebih banyak informasi tentang struktur data dan perhitungan yang sedang dilakukan. Secara internal, Spark SQL menggunakan informasi tambahan ini untuk melakukan pengoptimalan ekstra. Ada beberapa cara untuk berinteraksi dengan Spark SQL termasuk SQL dan Dataset API. Saat menghitung hasil mesin eksekusi yang sama digunakan, terlepas dari API / bahasa yang Anda gunakan untuk mengekspresikan perhitungan. Penyatuan ini berarti bahwa pengembang dapat dengan mudah beralih antara API yang berbeda berdasarkan cara yang paling alami untuk mengekspresikan transformasi tertentu.

Semua contoh pada halaman penggunaan data sampel ini termasuk dalam distribusi Spark dan dapat dijalankan di spark-shell, pysparkshell, atau sparkRshell.

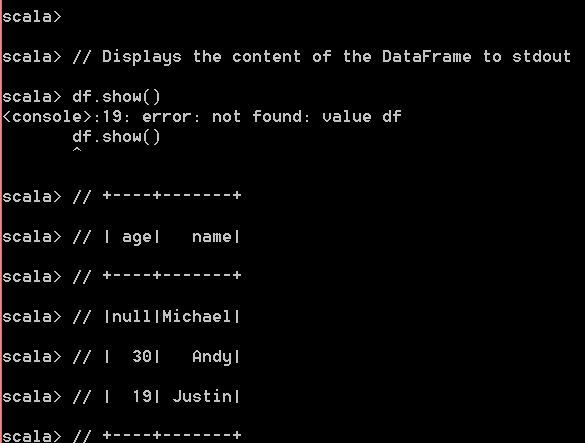
* Untuk masuk ke semua fungsi spark ada di sparksession class,dengan perintah sparksession.builder().



Hasil:

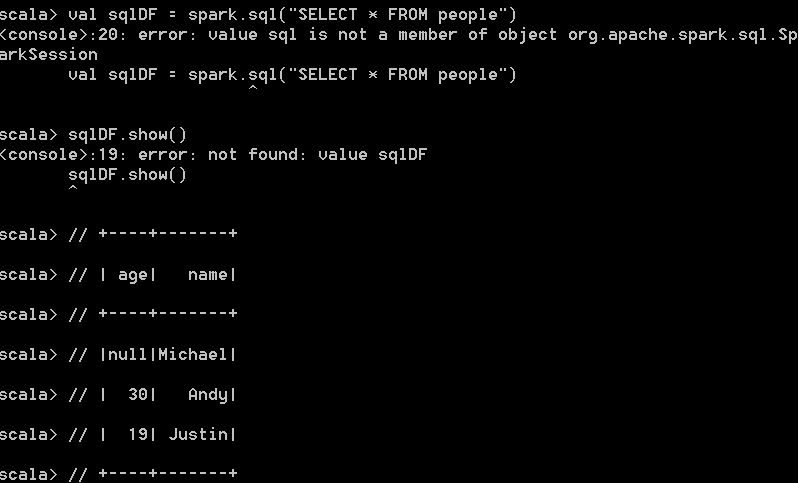


* Membuat data frame ,dengan sparksession aplikasi dapat membuat dataframe dari adaRDDD, table hive, atau dari sumber data spark



* Menjalankan query sql secara terprogram

Sql pada fungsi spark session memungkinkan aplikasi untuk menjalankan query sql pemograman dan mengembalikan hasilny sebagai dataframe



* Membuat datataset Dataset mirip dengan RDD, namun, alih-alih menggunakan serialisasi Java atau Kryo, mereka menggunakan [Encoder](https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.Encoder) khusus untuk membuat serialize objek untuk pemrosesan atau transmisi melalui jaringan. Sementara kedua encoders dan serialisasi standar bertanggung jawab untuk mengubah sebuah objek menjadi byte, encoders adalah kode yang dihasilkan secara dinamis dan menggunakan format yang memungkinkan Spark melakukan banyak operasi seperti penyaringan, sortasi dan hashing tanpa deserializing byte kembali ke objek.

*// Note: Case classes in Scala 2.10 can support only up to 22 fields. To work around this limit,*

*// you can use custom classes that implement the Product interface*

**case** **class** **Person**(name**:** String, age**:** Long)

*// Encoders are created for case classes*

**val** caseClassDS **=** **Seq**(**Person**("Andy", 32)).toDS()

caseClassDS.show()

*// +----+---+*

*// |name|age|*

*// +----+---+*

*// |Andy| 32|*

*// +----+---+*

*// Encoders for most common types are automatically provided by importing spark.implicits.\_*

**val** primitiveDS **=** **Seq**(1, 2, 3).toDS()

primitiveDS.map(**\_** + 1).collect() *// Returns: Array(2, 3, 4)*

*// DataFrames can be converted to a Dataset by providing a class. Mapping will be done by name*

**val** path **=** "examples/src/main/resources/people.json"

**val** peopleDS **=** spark.read.json(path).as[Person]

peopleDS.show()

*// +----+-------+*

*// | age| name|*

*// +----+-------+*

*// |null|Michael|*

*// | 30| Andy|*

*// | 19| Justin|*

*// +----+-------+*

