Spark SQL

Pregled

- Dataframe API
- SQL upiti
- Učitavanje i snimanje podataka
- Catalyst
- Tungsten
- DataSets

Dataframes

- RDD predstavlja low-level način manipulacije podacima u Spark-u
- Dataframe – strukturirani, distribuirani podaci predstavljeni u formi tabela
- Slični konceptu Dataframe iz Panda paketa za Python, ali su distrubuirani i optimizovani sa Catalyst-om

Dataframes 2

- Dataframes prevode SQL upite na optimizovane low-level operacije sa RDD-ovima, pa isti API može biti korišćen iz više različitih jezika (Python, Scala, Java, R)
- Jedan od najvažnijih koncepata u Spark okruženju
- U Spark 2.0 implementirani kao vrsta Dataset-ova

Kreiranje Dataframe-ova

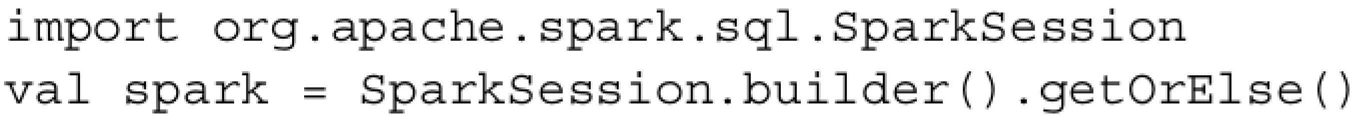
- Iz RDD-a
- SQL upitom
- Učitavanjem podataka iz spoljašnjeg izvora

RDD → Dataframe

- Učitavanje podataka inicijalno u RDD (load i transform sa ciljem strukturiranja podataka), a onda konvertovanje u Dataframe
- Tri načina
  - Sa RDD-om koji sadrži torke za svaki red
  - Case klase
  - Specificiranjem šeme

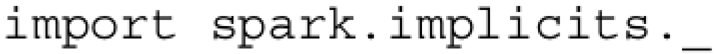
SparkSession

- Potrebno je u programu kreirati SparkSession objekat



```
import org.apache.spark.sql.SparkSession
val spark = SparkSession.builder().getOrElse()
```

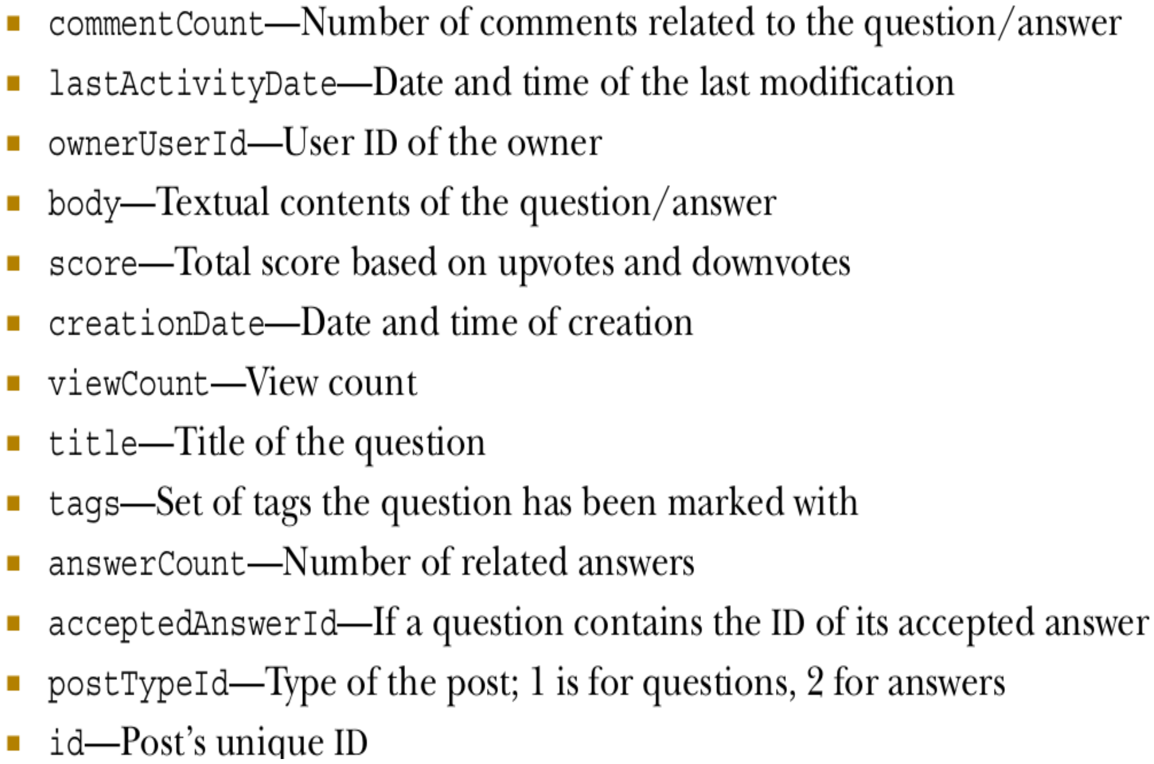
- Metode za automatsko konvertovanje RDD-a u Dataframe



```
import spark.implicits._
```

Primjer

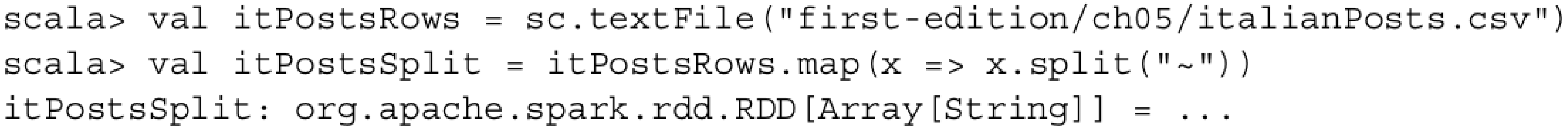
- U nastavku se koristi dataset sa Stack Exchange-a
- ch5/italianPosts.csv



- `commentCount`—Number of comments related to the question/answer
- `lastActivityDate`—Date and time of the last modification
- `ownerUserId`—User ID of the owner
- `body`—Textual contents of the question/answer
- `score`—Total score based on upvotes and downvotes
- `creationDate`—Date and time of creation
- `viewCount`—View count
- `title`—Title of the question
- `tags`—Set of tags the question has been marked with
- `answerCount`—Number of related answers
- `acceptedAnswerId`—If a question contains the ID of its accepted answer
- `postTypeId`—Type of the post; 1 is for questions, 2 for answers
- `id`—Post's unique ID

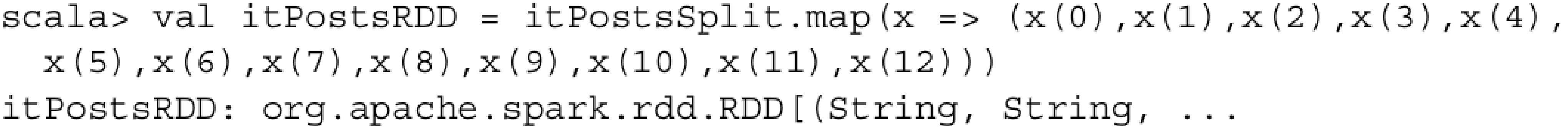
RDD sa torkama → Dataframe

- RDD koji sadrži niz stringova



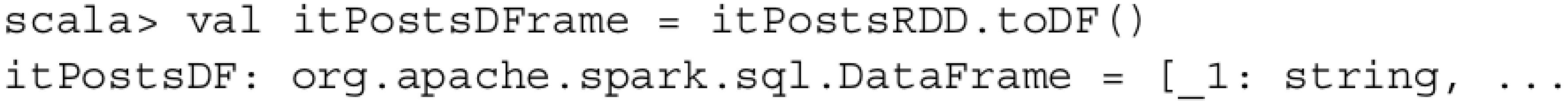
```
scala> val itPostsRows = sc.textFile("first-edition/ch05/italianPosts.csv")
scala> val itPostsSplit = itPostsRows.map(x => x.split("~"))
itPostsSplit: org.apache.spark.rdd.RDD[Array[String]] = ...
```

- RDD sa torkama



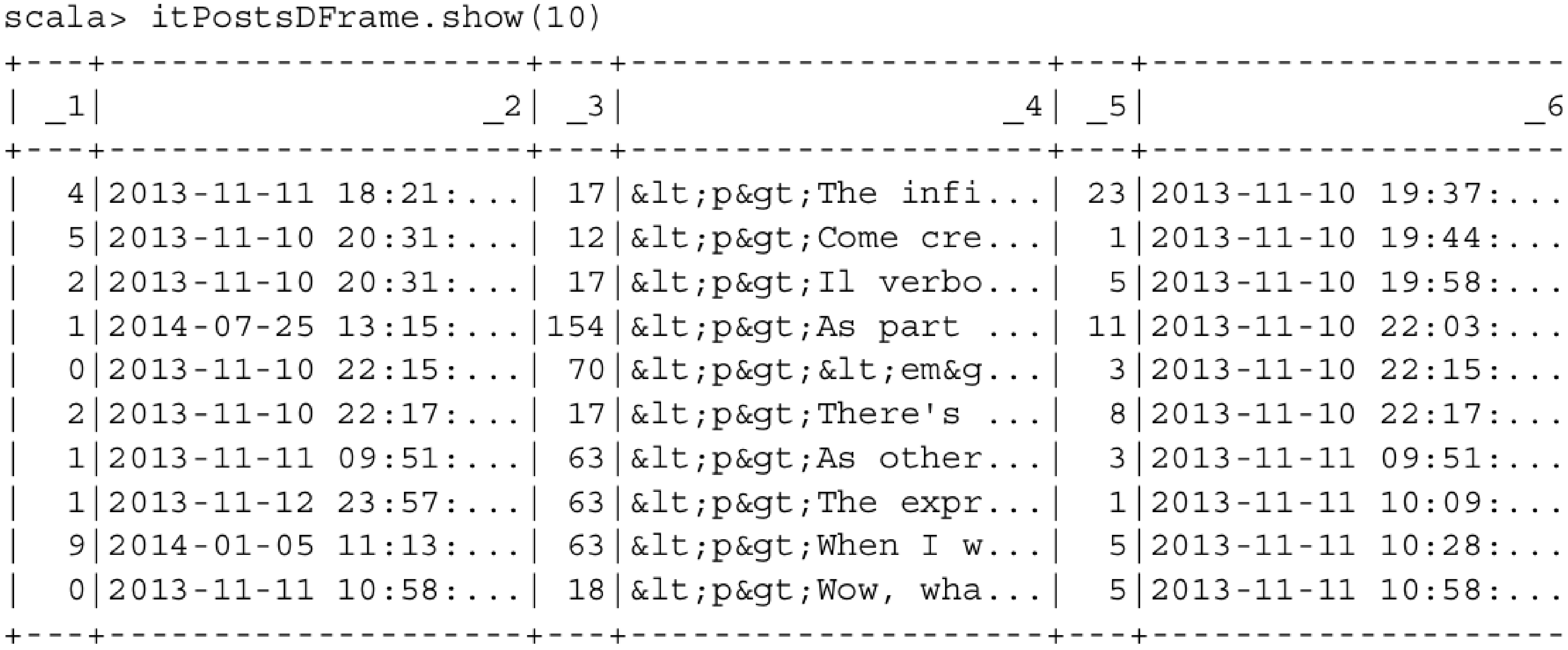
```
scala> val itPostsRDD = itPostsSplit.map(x => (x(0),x(1),x(2),x(3),x(4),
  x(5),x(6),x(7),x(8),x(9),x(10),x(11),x(12)))
itPostsRDD: org.apache.spark.rdd.RDD[(String, String, ...
```

- Funkcija toDF()



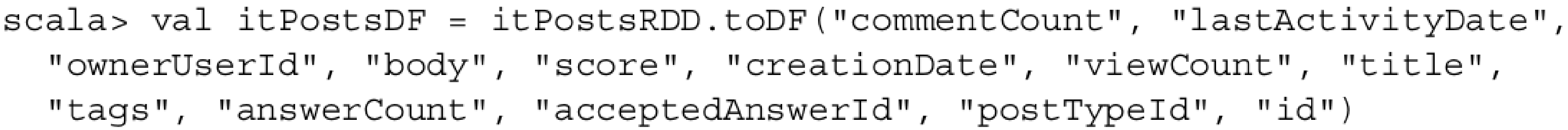
```
scala> val itPostsDFrame = itPostsRDD.toDF()
itPostsDF: org.apache.spark.sql.DataFrame = [_1: string, ...
```

toDF()

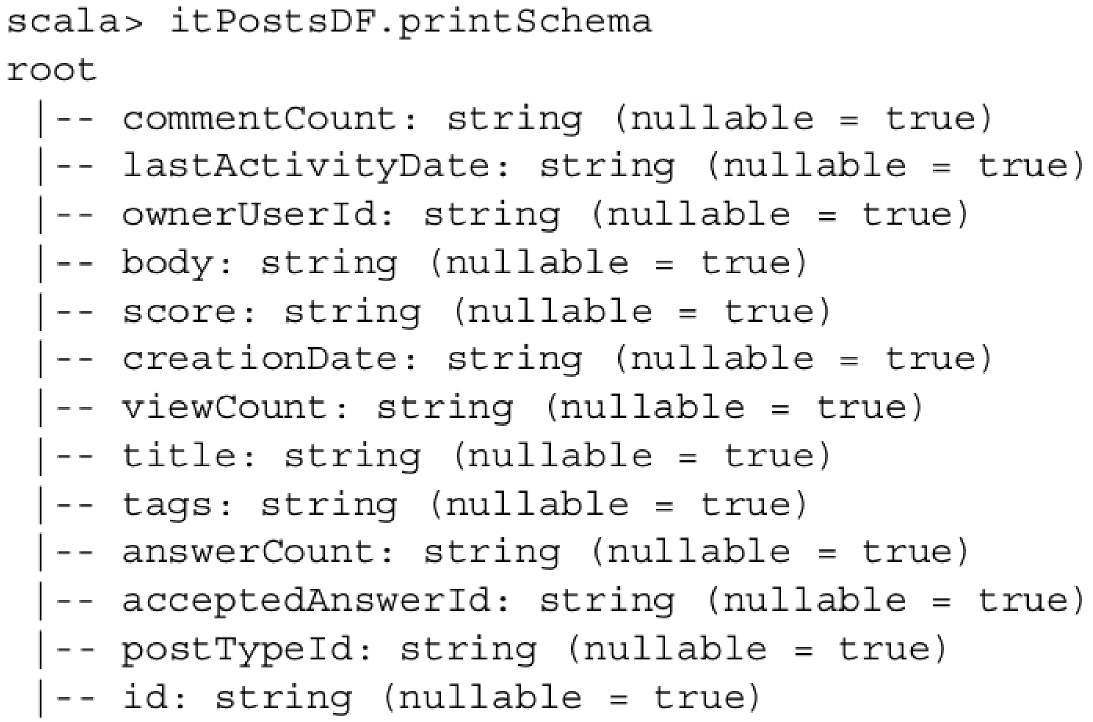


```
scala> itPostsDFrame.show(10)
+---+--------------------+---+--------------------+---+--------------------
| _1|                  _2| _3|                  _4| _5|                  _6
+---+--------------------+---+--------------------+---+--------------------
|  4|2013-11-11 18:21:...| 17|&lt;p&gt;The infi...| 23|2013-11-10 19:37:...
|  5|2013-11-10 20:31:...| 12|&lt;p&gt;Come cre...|  1|2013-11-10 19:44:...
|  2|2013-11-10 20:31:...| 17|&lt;p&gt;Il verbo...|  5|2013-11-10 19:58:...
|  1|2014-07-25 13:15:...|154|&lt;p&gt;As part ...| 11|2013-11-10 22:03:...
|  0|2013-11-10 22:15:...| 70|&lt;p&gt;&lt;em&g...|  3|2013-11-10 22:15:...
|  2|2013-11-10 22:17:...| 17|&lt;p&gt;There's ...|  8|2013-11-10 22:17:...
|  1|2013-11-11 09:51:...| 63|&lt;p&gt;As other...|  3|2013-11-11 09:51:...
|  1|2013-11-12 23:57:...| 63|&lt;p&gt;The expr...|  1|2013-11-11 10:09:...
|  9|2014-01-05 11:13:...| 63|&lt;p&gt;When I w...|  5|2013-11-11 10:28:...
|  0|2013-11-11 10:58:...| 18|&lt;p&gt;Wow, wha...|  5|2013-11-11 10:58:...
+---+--------------------+---+--------------------+---+--------------------
```

toDF() 2



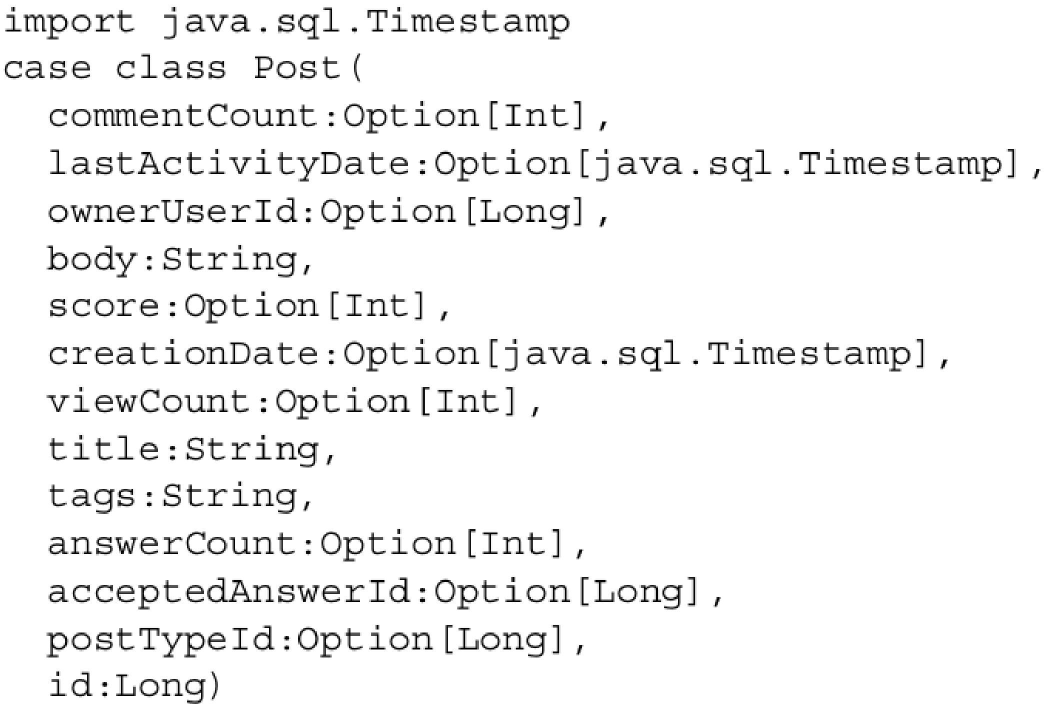
```
scala> val itPostsDF = itPostsRDD.toDF("commentCount", "lastActivityDate",
  "ownerUserId", "body", "score", "creationDate", "viewCount", "title",
  "tags", "answerCount", "acceptedAnswerId", "postTypeId", "id")
```



```
scala> itPostsDF.printSchema
root
 |-- commentCount: string (nullable = true)
 |-- lastActivityDate: string (nullable = true)
 |-- ownerUserId: string (nullable = true)
 |-- body: string (nullable = true)
 |-- score: string (nullable = true)
 |-- creationDate: string (nullable = true)
 |-- viewCount: string (nullable = true)
 |-- title: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- answerCount: string (nullable = true)
 |-- acceptedAnswerId: string (nullable = true)
 |-- postTypeId: string (nullable = true)
 |-- id: string (nullable = true)
```

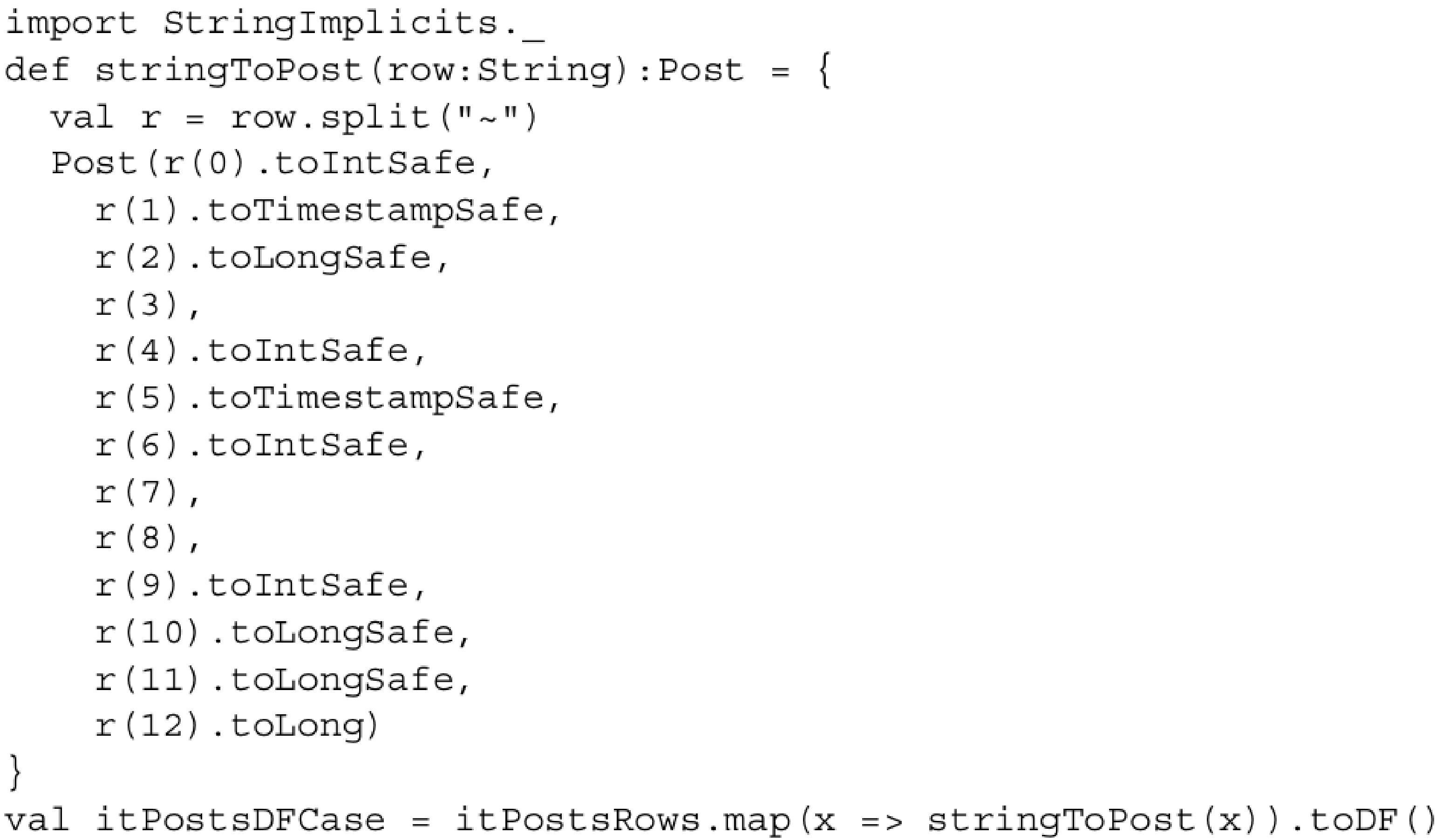
Case klase

- Mapiranje svakog reda iz RDD-a na case klasu u poziv funkcije toDF()
- NULL polja - Option[T]



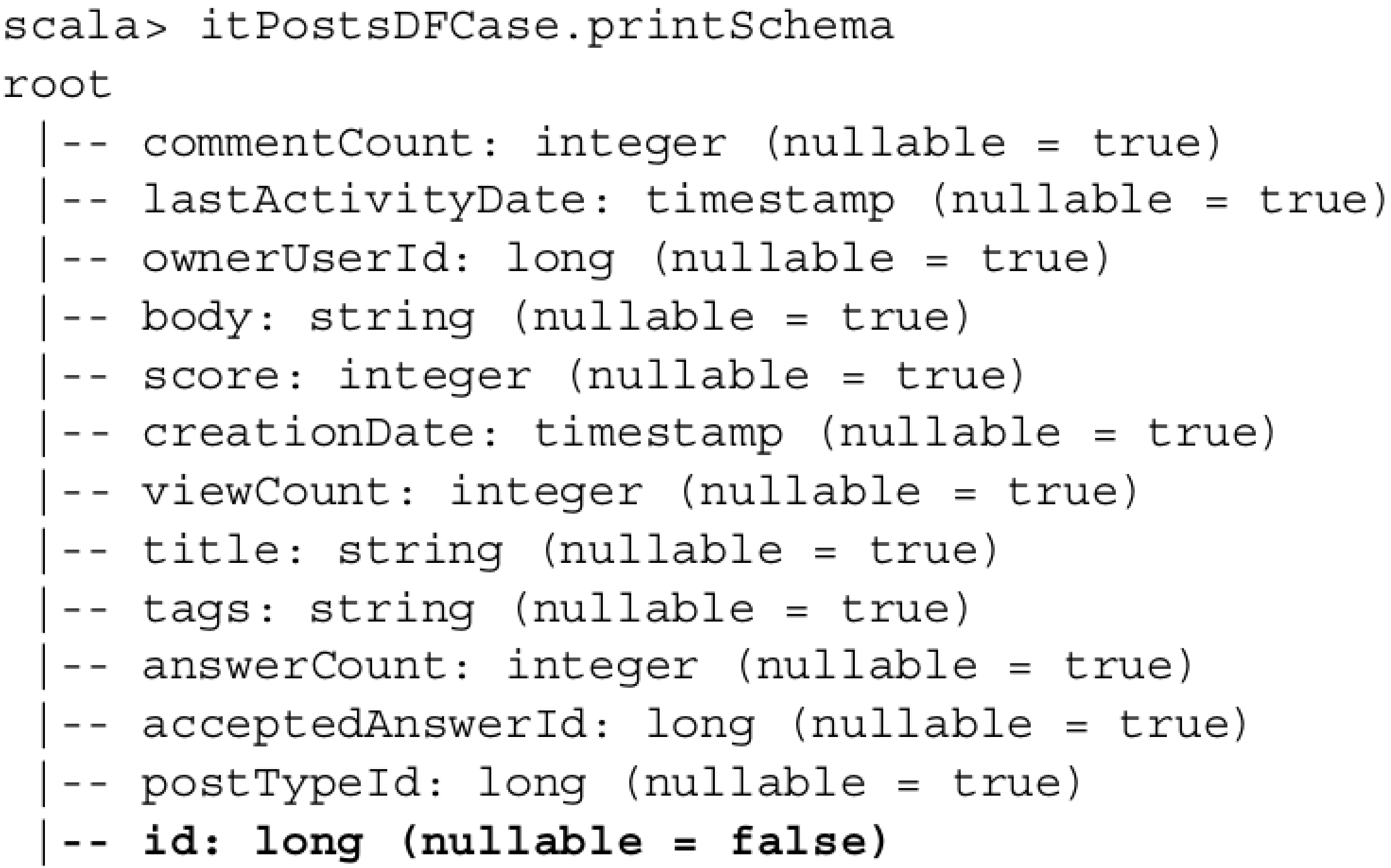
```
import java.sql.Timestamp
case class Post(
  commentCount:Option[Int],
  lastActivityDate:Option[java.sql.Timestamp],
  ownerUserId:Option[Long],
  body:String,
  score:Option[Int],
  creationDate:Option[java.sql.Timestamp],
  viewCount:Option[Int],
  title:String,
  tags:String,
  answerCount:Option[Int],
  acceptedAnswerId:Option[Long],
  postTypeId:Option[Long],
  id:Long)
```

Case klase 2



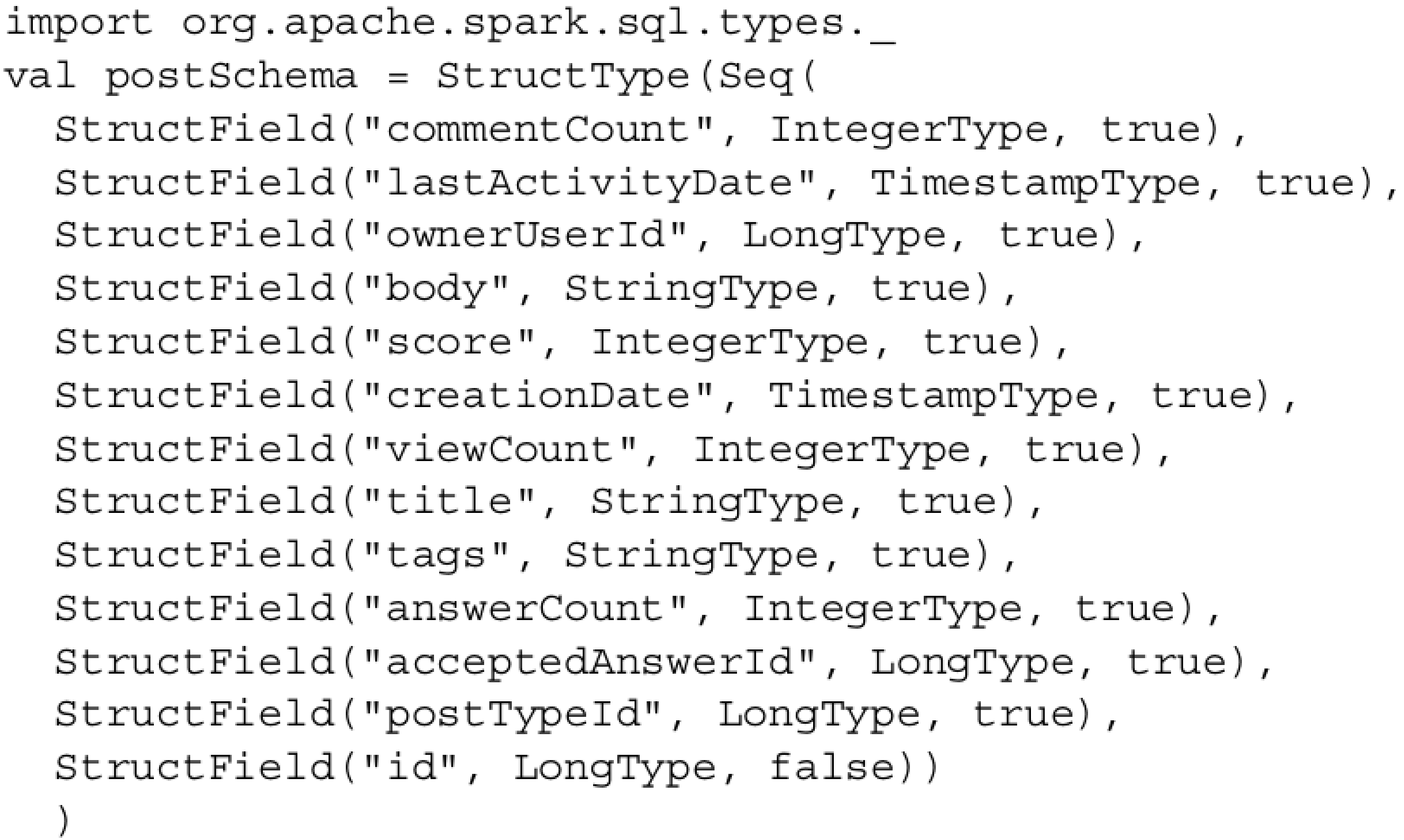
```
import StringImplicits._
def stringToPost(row:String):Post = {
  val r = row.split("~")
  Post(r(0).toIntSafe,
    r(1).toTimestampSafe,
    r(2).toLongSafe,
    r(3),
    r(4).toIntSafe,
    r(5).toTimestampSafe,
    r(6).toIntSafe,
    r(7),
    r(8),
    r(9).toIntSafe,
    r(10).toLongSafe,
    r(11).toLongSafe,
    r(12).toLong)
}
val itPostsDFCase = itPostsRows.map(x => stringToPost(x)).toDF()
```

Case klase 3



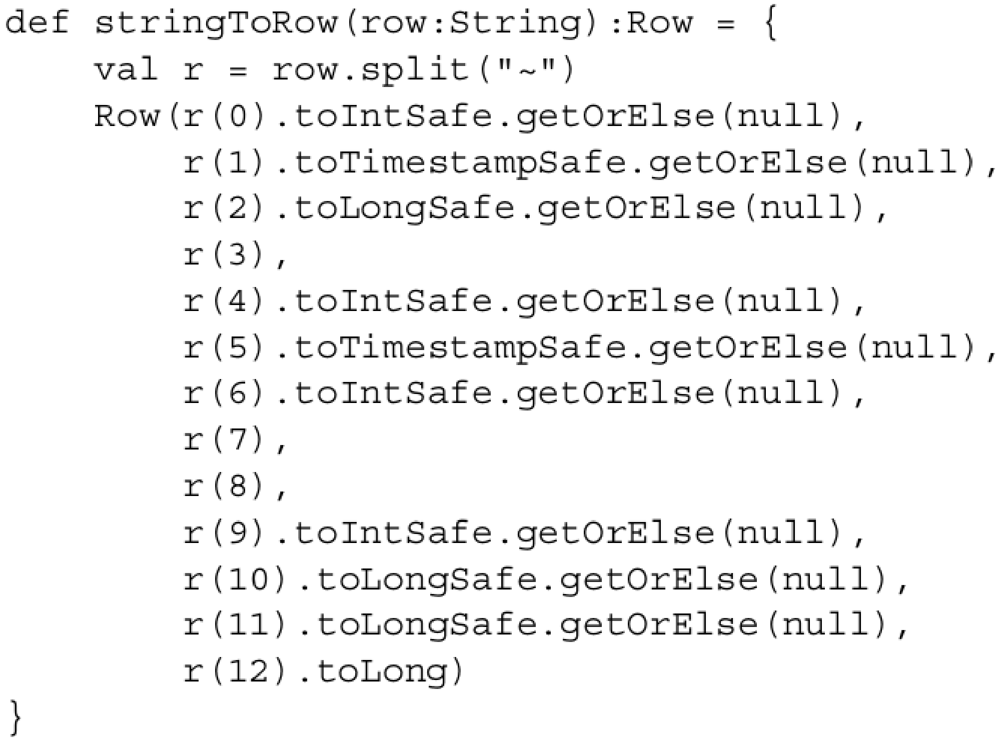
```
scala> itPostsDFCase.printSchema
root
 |-- commentCount: integer (nullable = true)
 |-- lastActivityDate: timestamp (nullable = true)
 |-- ownerUserId: long (nullable = true)
 |-- body: string (nullable = true)
 |-- score: integer (nullable = true)
 |-- creationDate: timestamp (nullable = true)
 |-- viewCount: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- answerCount: integer (nullable = true)
 |-- acceptedAnswerId: long (nullable = true)
 |-- postTypeId: long (nullable = true)
 |-- id: long (nullable = false)
```

Specifikovanie šeme

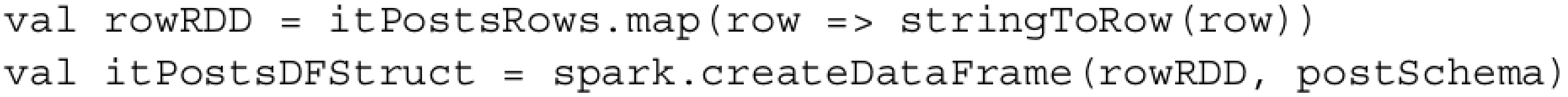


```
import org.apache.spark.sql.types._
val postSchema = StructType(Seq(
  StructField("commentCount", IntegerType, true),
  StructField("lastActivityDate", TimestampType, true),
  StructField("ownerUserId", LongType, true),
  StructField("body", StringType, true),
  StructField("score", IntegerType, true),
  StructField("creationDate", TimestampType, true),
  StructField("viewCount", IntegerType, true),
  StructField("title", StringType, true),
  StructField("tags", StringType, true),
  StructField("answerCount", IntegerType, true),
  StructField("acceptedAnswerId", LongType, true),
  StructField("postTypeId", LongType, true),
  StructField("id", LongType, false))
  )
```

createDataFrame



```
def stringToRow(row:String):Row = {
    val r = row.split("~")
    Row(r(0).toIntSafe.getOrElse(null),
        r(1).toTimestampSafe.getOrElse(null),
        r(2).toLongSafe.getOrElse(null),
        r(3),
        r(4).toIntSafe.getOrElse(null),
        r(5).toTimestampSafe.getOrElse(null),
        r(6).toIntSafe.getOrElse(null),
        r(7),
        r(8),
        r(9).toIntSafe.getOrElse(null),
        r(10).toLongSafe.getOrElse(null),
        r(11).toLongSafe.getOrElse(null),
        r(12).toLong)
}
```



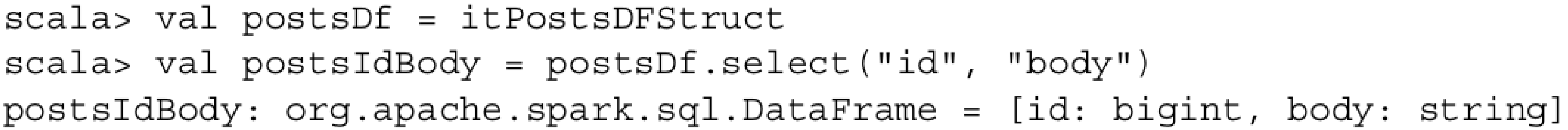
```
val rowRDD = itPostsRows.map(row => stringToRow(row))
val itPostsDFStruct = spark.createDataFrame(rowRDD, postSchema)
```

Basic API

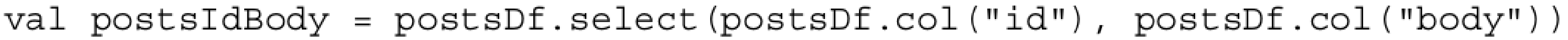
- Basic API
  - Select, filter, map, group, join
- Dataframes
  - immutable
  - lazy

Selecting

- Projekcija na specifikovane kolone, kolone se zadaju preko imena ili kao Column objekti

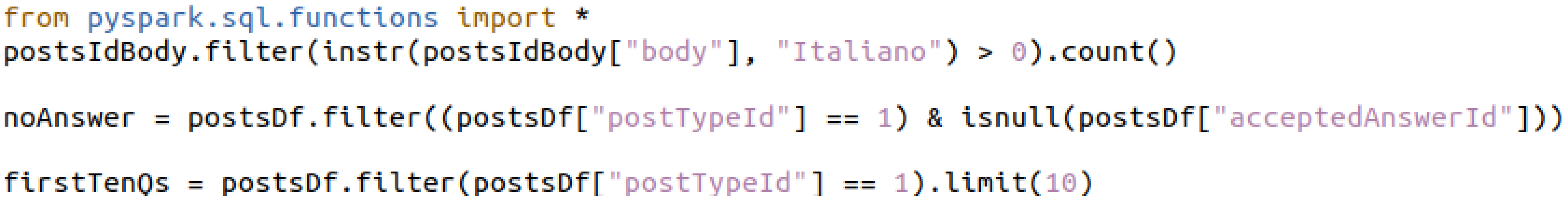


```
scala> val postsDf = itPostsDFStruct
scala> val postsIdBody = postsDf.select("id", "body")
postsIdBody: org.apache.spark.sql.DataFrame = [id: bigint, body: string]
```



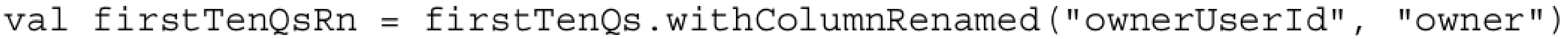
```
val postsIdBody = postsDf.select(postsDf.col("id"), postsDf.col("body"))
```

Filtering

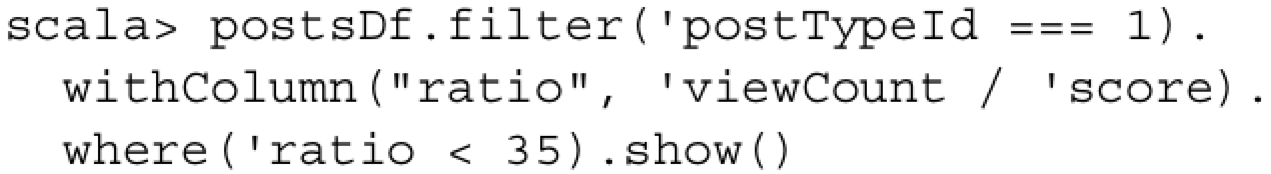


```
from pyspark.sql.functions import *
postsIdBody.filter(instr(postsIdBody["body"], "Italiano") > 0).count()

noAnswer = postsDf.filter((postsDf["postTypeId"] == 1) & isnull(postsDf["acceptedAnswerId"]))

firstTenQs = postsDf.filter(postsDf["postTypeId"] == 1).limit(10)
```

Preimenovanje i dodavanje kolone



```
val firstTenQsRn = firstTenQs.withColumnRenamed("ownerUserId", "owner")
```



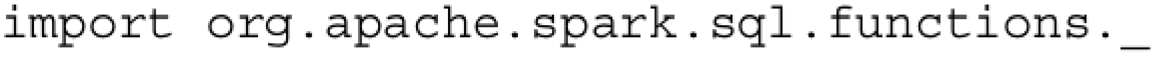
```
scala> postsDf.filter('postTypeId === 1).
  withColumn("ratio", 'viewCount / 'score).
  where('ratio < 35).show()
```

SQL funkcije

- Skalarne funkcije
- Agregatne funkcije
- Window funkcije
  - Vraćaju više vrijednosti za grupu redova
- User-defined funkcije UDF

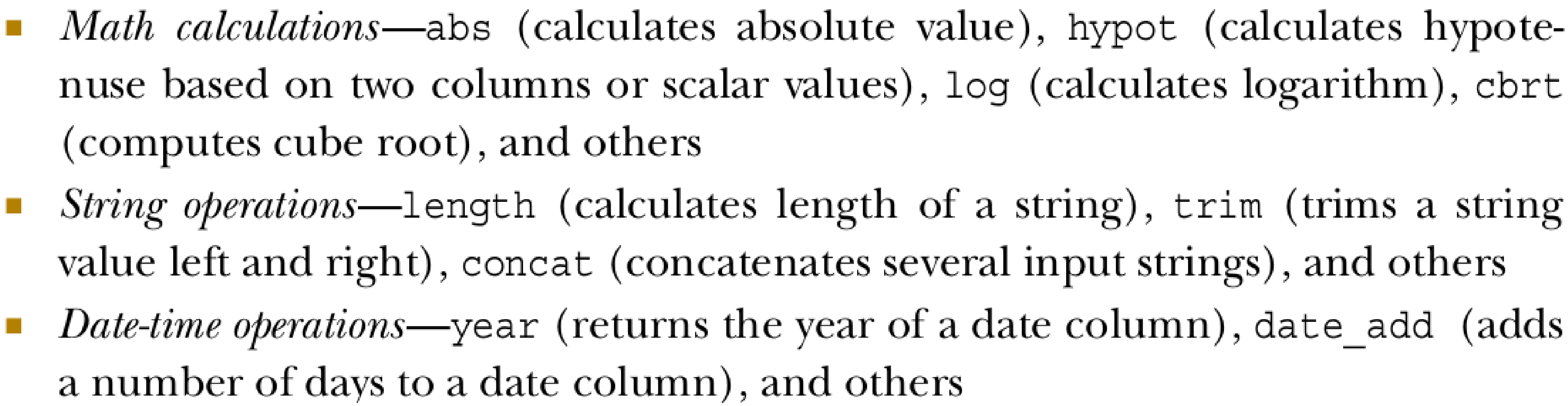
Skalarne i agregatne funkcije

- Skalarne funkcije vraćaju jednu vrijednost za svaki red na osnovu vrijednosti jedni ili više kolona
- Agregatne funkcije vraćaju jednu vrijednost za grupu redova (u kombinaciji sa groupBy)
  - min, max, count, avg, sum



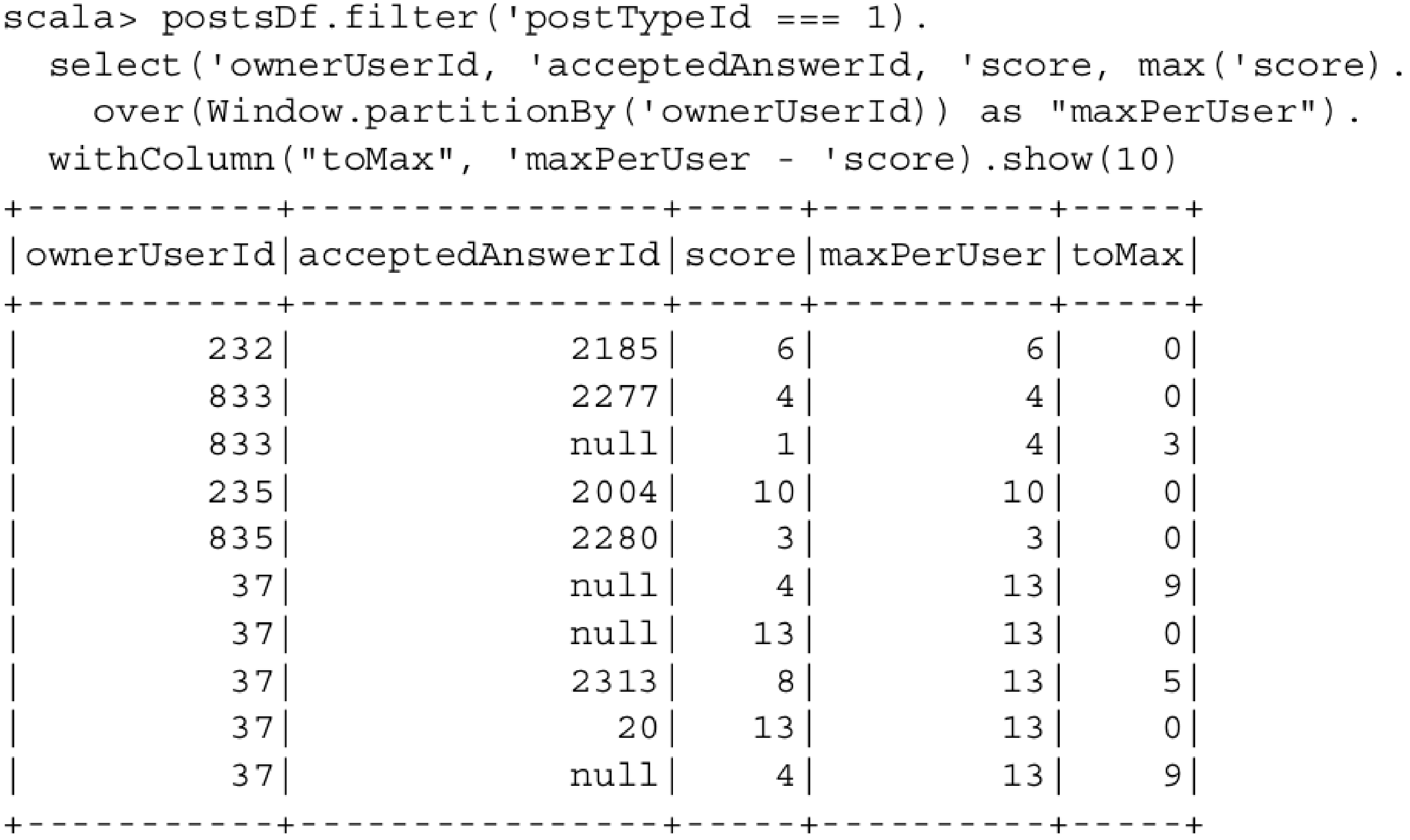
```
import org.apache.spark.sql.functions._
```

Primjeri skalarnih funkcija



- *Math calculations*—`abs` (calculates absolute value), `hypot` (calculates hypotenuse based on two columns or scalar values), `log` (calculates logarithm), `cbrt` (computes cube root), and others
- *String operations*—`length` (calculates length of a string), `trim` (trims a string value left and right), `concat` (concatenates several input strings), and others
- *Date-time operations*—`year` (returns the year of a date column), `date_add` (adds a number of days to a date column), and others

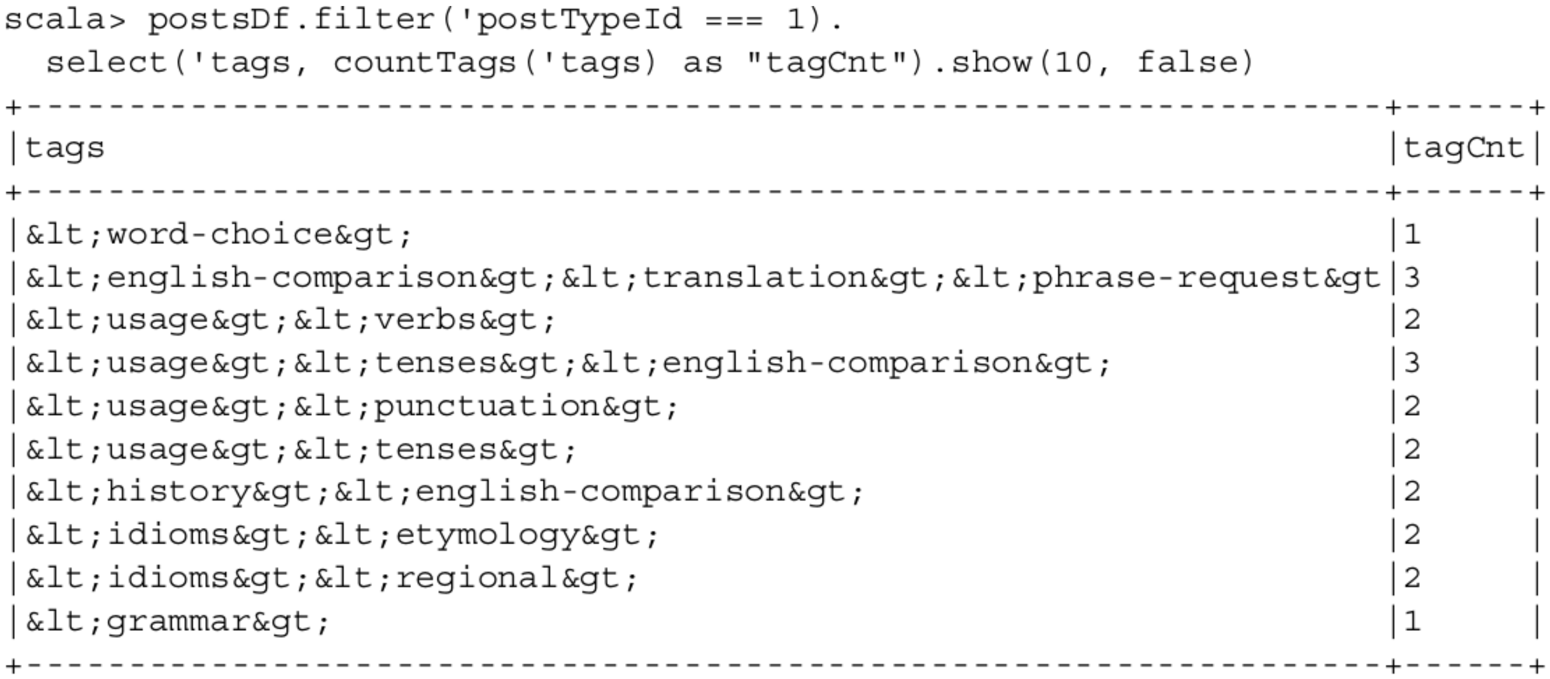
Window funkcije



```
scala> postsDf.filter('postTypeId === 1).
  select('ownerUserId, 'acceptedAnswerId, 'score, max('score).
    over(Window.partitionBy('ownerUserId)) as "maxPerUser").
  withColumn("toMax", 'maxPerUser - 'score).show(10)
+-----------+----------------+-----+----------+-----+
|ownerUserId|acceptedAnswerId|score|maxPerUser|toMax|
+-----------+----------------+-----+----------+-----+
|        232|            2185|    6|         6|    0|
|        833|            2277|    4|         4|    0|
|        833|            null|    1|         4|    3|
|        235|            2004|   10|        10|    0|
|        835|            2280|    3|         3|    0|
|         37|            null|    4|        13|    9|
|         37|            null|   13|        13|    0|
|         37|            2313|    8|        13|    5|
|         37|              20|   13|        13|    0|
|         37|            null|    4|        13|    9|
+-----------+----------------+-----+----------+-----+
```

UDFs

```
scala> val countTags = udf((tags: String) =>
  "&lt;".r.findAllMatchIn(tags).length)
countTags: org.apache.spark.sql.UserDefinedFunction = ...
```



```
scala> postsDf.filter('postTypeId === 1).
  select('tags, countTags('tags) as "tagCnt").show(10, false)
+------------------------------------------------------------------+------+
|tags                                                              |tagCnt|
+------------------------------------------------------------------+------+
|&lt;word-choice&gt;                                              |1     |
|&lt;english-comparison&gt;&lt;translation&gt;&lt;phrase-request&gt|3     |
|&lt;usage&gt;&lt;verbs&gt;                                      |2     |
|&lt;usage&gt;&lt;tenses&gt;&lt;english-comparison&gt;          |3     |
|&lt;usage&gt;&lt;punctuation&gt;                                |2     |
|&lt;usage&gt;&lt;tenses&gt;                                     |2     |
|&lt;history&gt;&lt;english-comparison&gt;                       |2     |
|&lt;idioms&gt;&lt;etymology&gt;                                 |2     |
|&lt;idioms&gt;&lt;regional&gt;                                  |2     |
|&lt;grammar&gt;                                                  |1     |
+------------------------------------------------------------------+------+
```

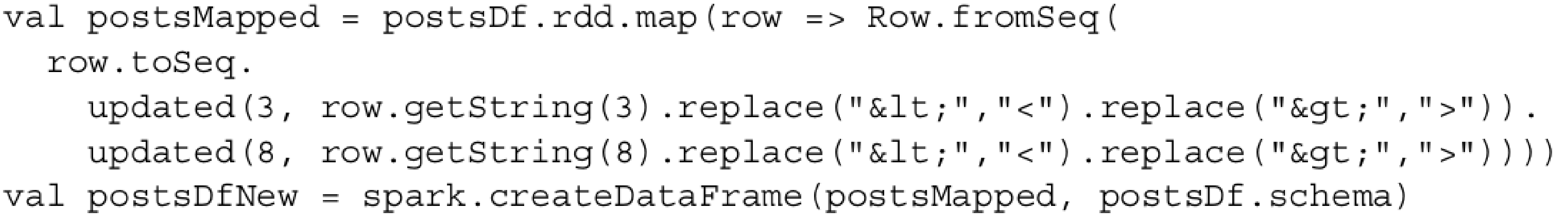
Missing vrijednosti

- drop – briše redove koji sadrže null ili NaN u bar jednoj koloni
- drop(“col”) – briše redove koji sadrže null ili NaN u koloni col
- fill - postsDf.na.fill(Map("viewCount" -> 0))

Dataframe → RDD

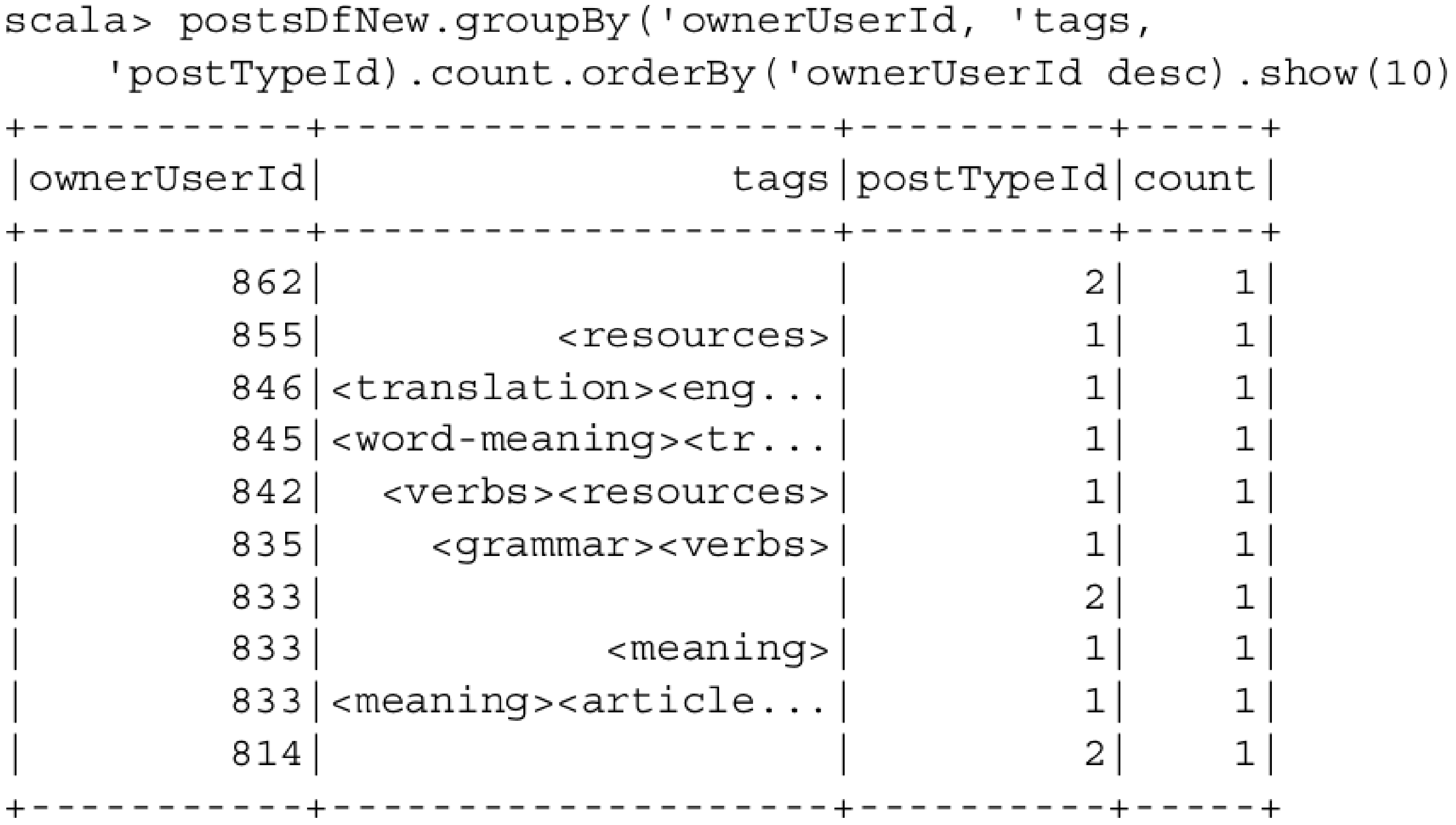
- rdd polje Dataframe objekta sadrži bazni RDD objekat sa elementima tipa Row
- Row ima razne getere: getString(index), getInt(index), getMap(index)
- Poziv metoda map, flatMap, Mappartitions nad Dataframe objektom odnosi se na njegovo rdd polje
  - Nije moguće automatski konvertovati u Dataframe rezultat poziva ovih metoda

Dataframe → RDD



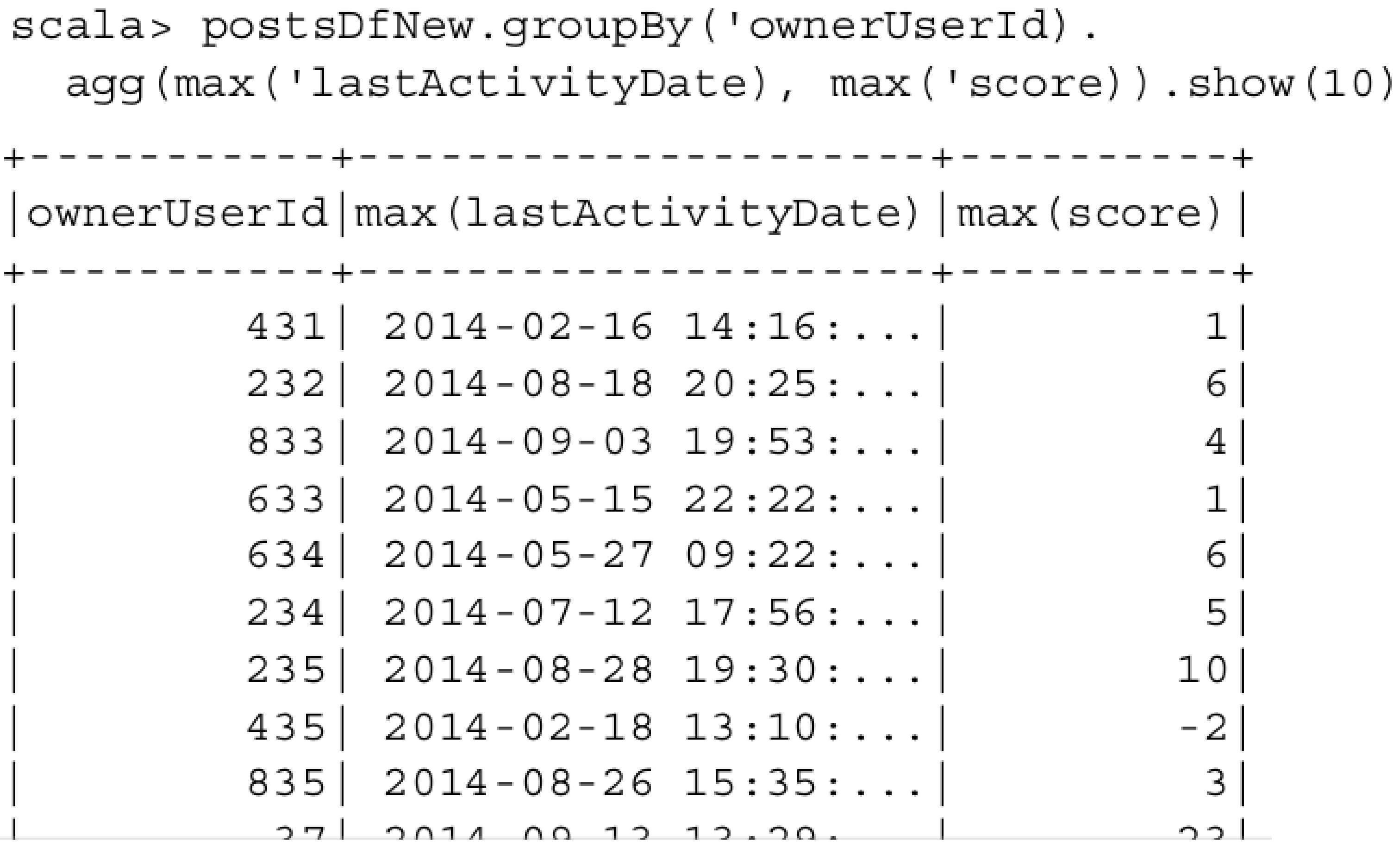
```
val postsMapped = postsDf.rdd.map(row => Row.fromSeq(
  row.toSeq.
    updated(3, row.getString(3).replace("&lt;","<").replace("&gt;",">")).
    updated(8, row.getString(8).replace("&lt;","<").replace("&gt;",">"))))
val postsDfNew = spark.createDataFrame(postsMapped, postsDf.schema)
```

Grupisanje



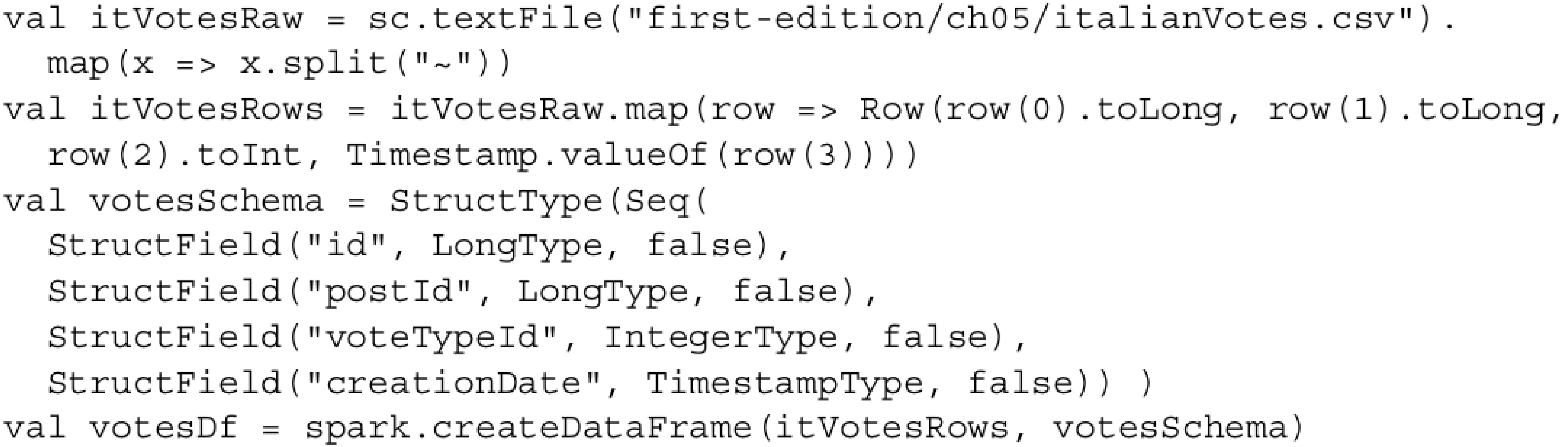
```
scala> postsDfNew.groupBy('ownerUserId, 'tags,
    'postTypeId).count.orderBy('ownerUserId desc).show(10)
+-----------+--------------------+----------+-----+
|ownerUserId|                tags|postTypeId|count|
+-----------+--------------------+----------+-----+
|        862|                    |         2|    1|
|        855|         <resources>|         1|    1|
|        846|<translation><eng...|         1|    1|
|        845|<word-meaning><tr...|         1|    1|
|        842|  <verbs><resources>|         1|    1|
|        835|    <grammar><verbs>|         1|    1|
|        833|                    |         2|    1|
|        833|           <meaning>|         1|    1|
|        833|<meaning><article...|         1|    1|
|        814|                    |         2|    1|
+-----------+--------------------+----------+-----+
```

Grupisanje 2

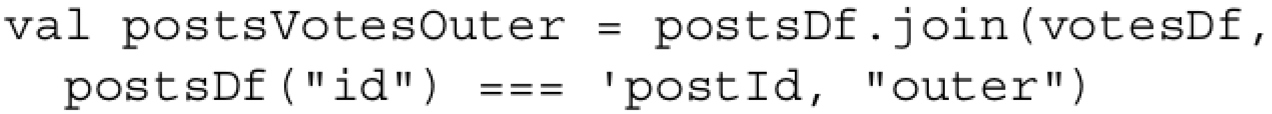


```
scala> postsDfNew.groupBy('ownerUserId).
  agg(max('lastActivityDate), max('score)).show(10)

+-----------+--------------------+----------+
|ownerUserId|max(lastActivityDate)|max(score)|
+-----------+--------------------+----------+
|        431| 2014-02-16 14:16:...|         1|
|        232| 2014-08-18 20:25:...|         6|
|        833| 2014-09-03 19:53:...|         4|
|        633| 2014-05-15 22:22:...|         1|
|        634| 2014-05-27 09:22:...|         6|
|        234| 2014-07-12 17:56:...|         5|
|        235| 2014-08-28 19:30:...|        10|
|        435| 2014-02-18 13:10:...|        -2|
|        835| 2014-08-26 15:35:...|         3|
```

Spajanje



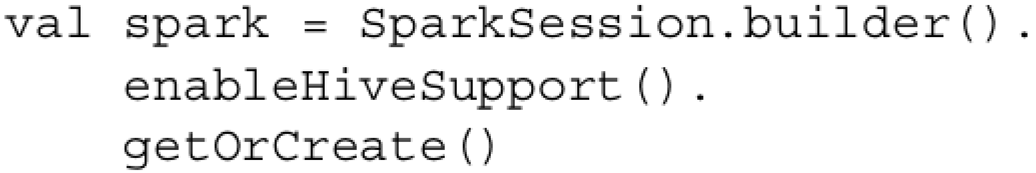
```
val itVotesRaw = sc.textFile("first-edition/ch05/italianVotes.csv").
  map(x => x.split("~"))
val itVotesRows = itVotesRaw.map(row => Row(row(0).toLong, row(1).toLong,
  row(2).toInt, Timestamp.valueOf(row(3))))
val votesSchema = StructType(Seq(
  StructField("id", LongType, false),
  StructField("postId", LongType, false),
  StructField("voteTypeId", IntegerType, false),
  StructField("creationDate", TimestampType, false)) )
val votesDf = spark.createDataFrame(itVotesRows, votesSchema)
```



```
val postsVotesOuter = postsDf.join(votesDf,
  postsDf("id") === 'postId, "outer")
```

SQL upiti

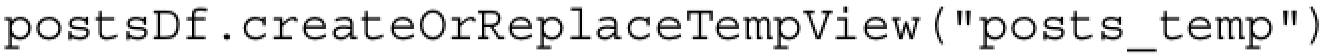
- Thrift - iz aplikacija koristeći JDBC ili ODBC protokole
- SQL dijalekti: Spark SQL i Hive Query Language



```
val spark = SparkSession.builder().
    enableHiveSupport().
    getOrCreate()
```

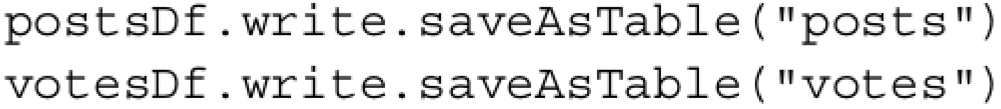
Katalog tabela

- Registrovanje Dataframe-a kao tabele sa imenom
  - privremeno



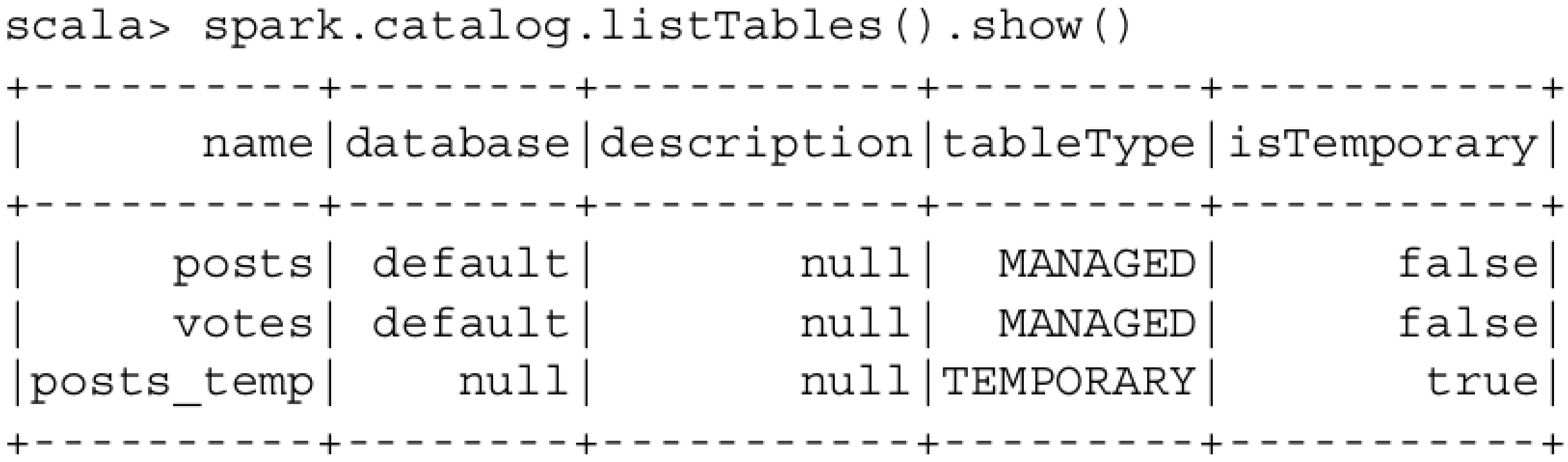
```
postsDf.createOrReplaceTempView("posts_temp")
```

  - u Hive metastore (perzistentna baza)



```
postsDf.write.saveAsTable("posts")
votesDf.write.saveAsTable("votes")
```

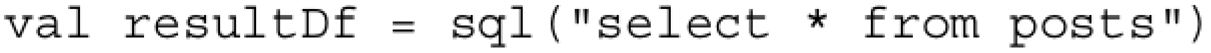
Katalog tabela 2



```
scala> spark.catalog.listTables().show()
+----------+--------+-----------+---------+-----------+
|      name|database|description|tableType|isTemporary|
+----------+--------+-----------+---------+-----------+
|     posts| default|       null|  MANAGED|      false|
|     votes| default|       null|  MANAGED|      false|
|posts_temp|    null|       null|TEMPORARY|       true|
+----------+--------+-----------+---------+-----------+
```

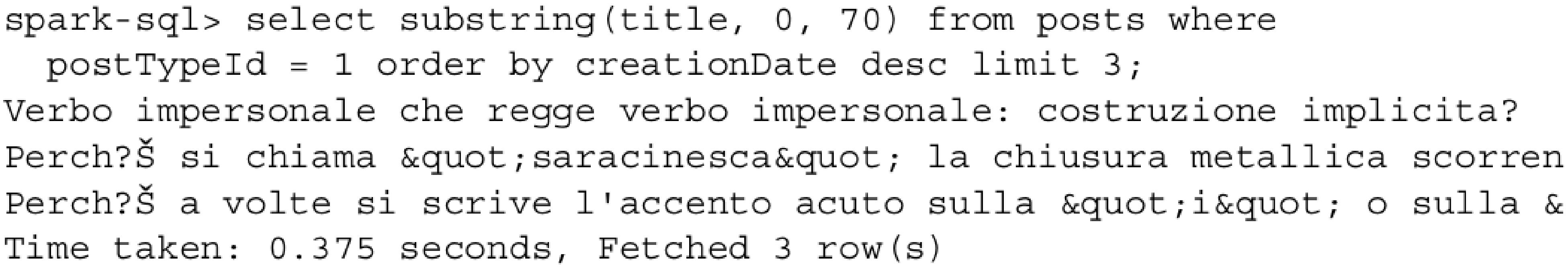
Izvršavanje upita

- import spark.sql



```
val resultDf = sql("select * from posts")
```

- Spark-sql shell

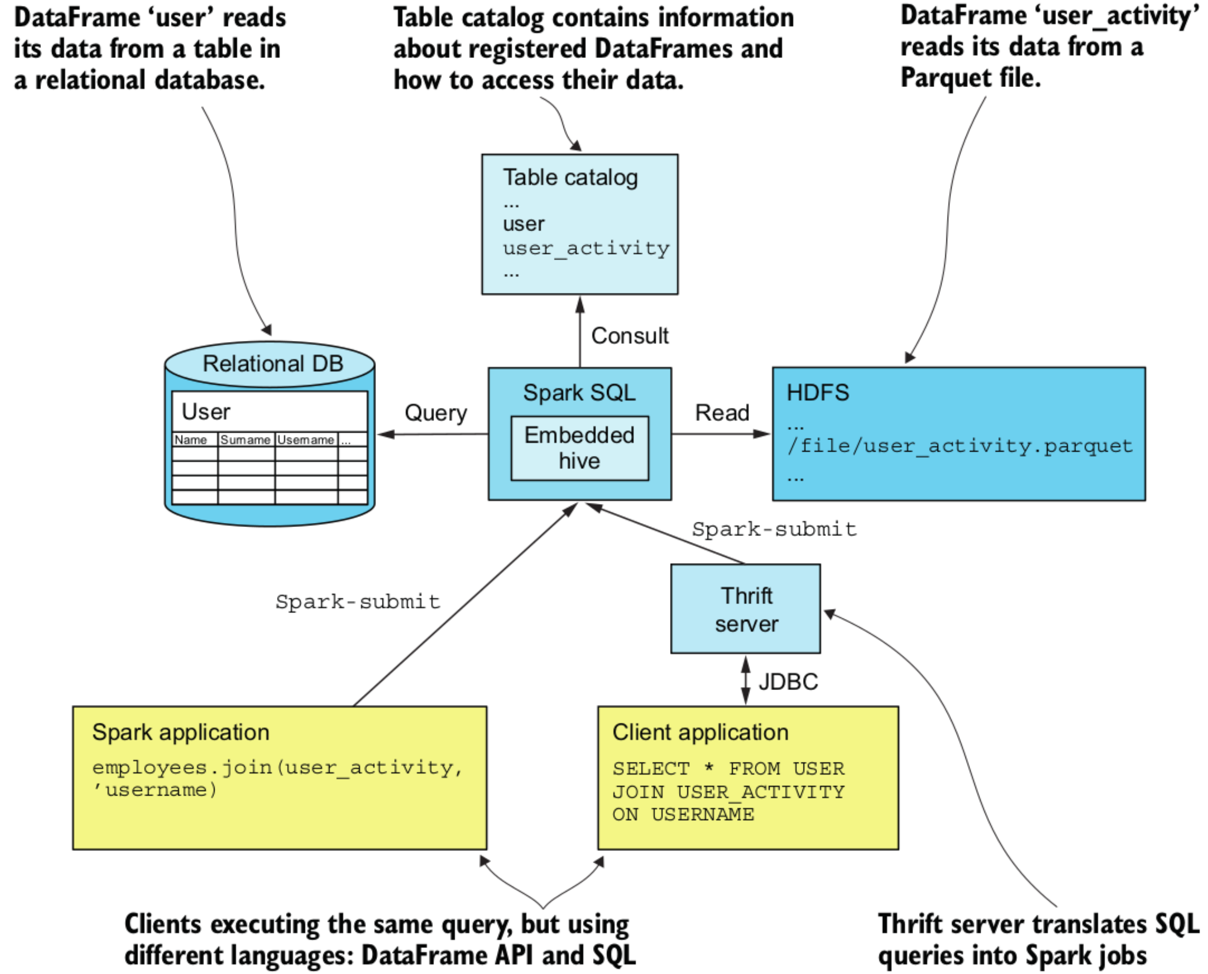


```
spark-sql> select substring(title, 0, 70) from posts where
  postTypeId = 1 order by creationDate desc limit 3;
Verbo impersonale che regge verbo impersonale: costruzione implicita?
Perch?Š si chiama &quot;saracinesca&quot; la chiusura metallica scorren
Perch?Š a volte si scrive l'accento acuto sulla &quot;i&quot; o sulla &
Time taken: 0.375 seconds, Fetched 3 row(s)
```

Thrift server

- Thrift server “otvara” Spark za aplikacije koje komuniciraju sa relacionim bazama JDBC ili ODBC protokolima
- Upiti se transformiš u Dataframe, odnosno RDD operacije a rezultat vraća preko JDBC/ODBC protokola

Thrift server



DataFrame ‘user’ reads its data from a table in a relational database.

Table catalog contains information about registered DataFrames and how to access their data.

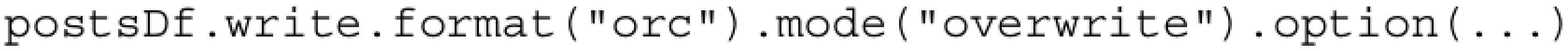
DataFrame ‘user_activity’ reads its data from a Parquet file.

Clients executing the same query, but using different languages: DataFrame API and SQL

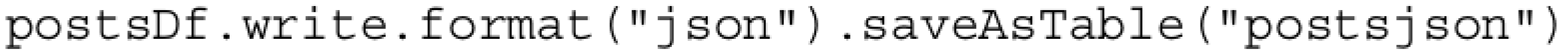
Thrift server translates SQL queries into Spark jobs

Snimanje dataframe-a

- Built-in datasources
  - JSON
  - Optimized row columnar – ORC
  - Parquet



```
postsDf.write.format("orc").mode("overwrite").option(...)
```



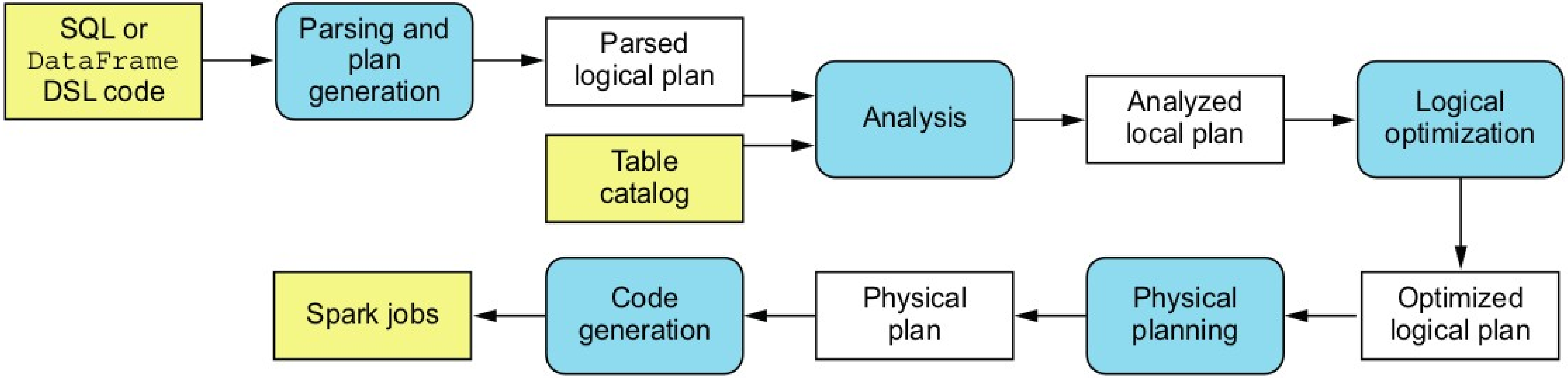
```
postsDf.write.format("json").saveAsTable("postsjson")
```

Učitavanje dataframe-a

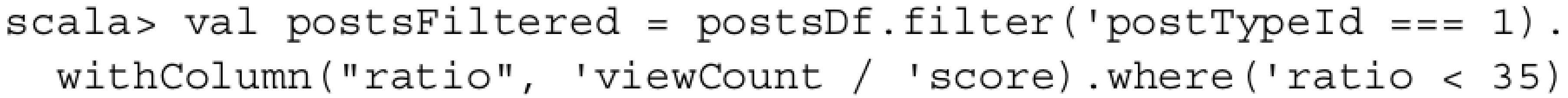
- org.apache.spark.sql.DataFrameReader
- Metod load (json, orc, parquet itd.)
- postsDf = spark.read.table("posts")

Catalyst

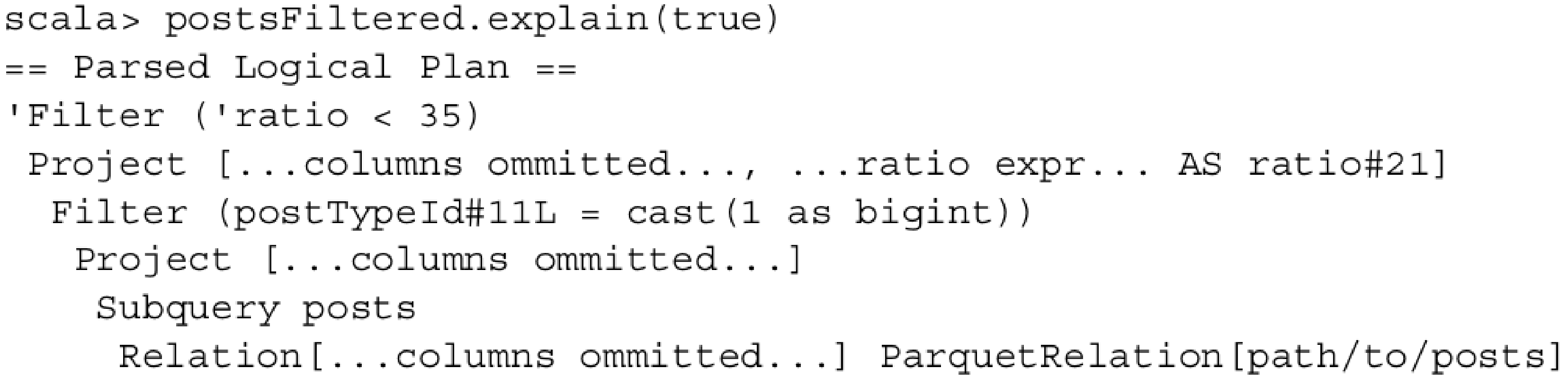
- Konvertuje Dataframe operacije na RDD



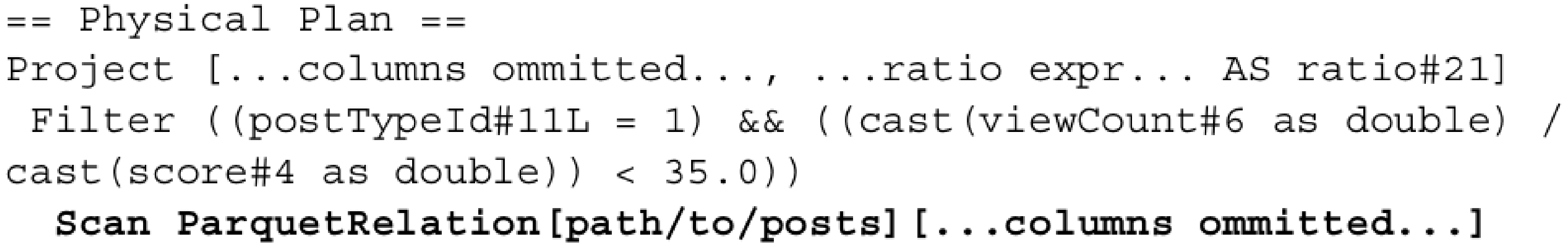
Primjer



```
scala> val postsFiltered = postsDf.filter('postTypeId === 1).
  withColumn("ratio", 'viewCount / 'score).where('ratio < 35)
```



```
scala> postsFiltered.explain(true)
== Parsed Logical Plan ==
'Filter ('ratio < 35)
 Project [...columns ommitted..., ...ratio expr... AS ratio#21]
  Filter (postTypeId#11L = cast(1 as bigint))
   Project [...columns ommitted...]
    Subquery posts
     Relation[...columns ommitted...] ParquetRelation[path/to/posts]
```



```
== Physical Plan ==
Project [...columns ommitted..., ...ratio expr... AS ratio#21]
 Filter ((postTypeId#11L = 1) && ((cast(viewCount#6 as double) /
cast(score#4 as double)) < 35.0))
  Scan ParquetRelation[path/to/posts][...columns ommitted...]
```

Tungsten

- Unapređenja kod
  - Upravljanja memorijom (binary encoded objects)
  - Sortiranje
  - Agregiranje
  - Shuffling