## WEEK 1 : Big Data SQL: Hive

## WEEK 2 : Big Data SQL: Hive (practice week)

## **WEEK 3 : Spark SQL and Spark Dataframe**

**Learning Objectives**

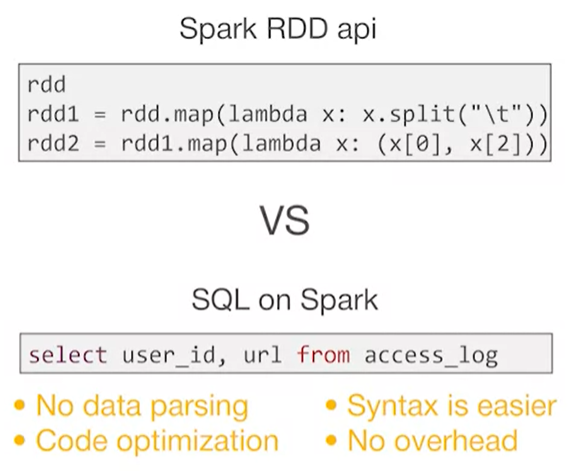
is dedicated to data analysis using Spark SQL. And at the end of the week you will be able to read, write and process structured data in different ways by Spark SQL. This week contains three lessons.

* First lesson, we will discuss the general properties of Spark SQL. In particular, why it's needed, how it works, what opportunities it has?
* Second lesson will be devoted to the basic method of data processing. There you will learn how to extract information from our data, how to build aggregates, how to join tables and process them with different functions.
* Third lesson, you will figure out how to handle missing queries, date and time, set your own functions, use Windows and you'll do many other interesting things.

### **3.1 Introducing DataFrame and SQL**

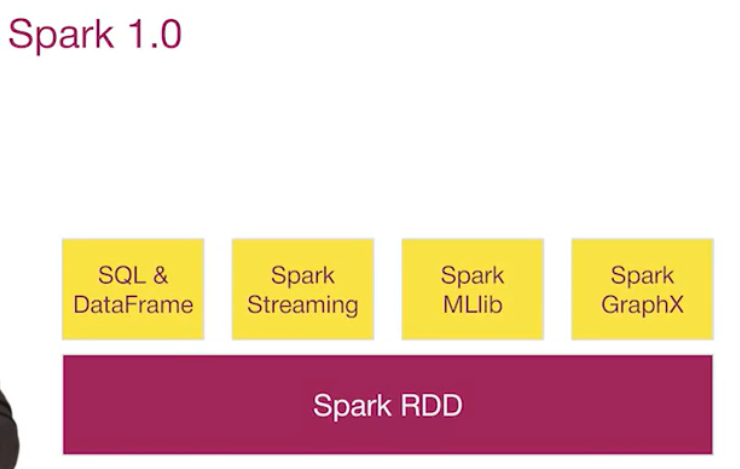
#### **3.1.1 Advantages of Spark SQL**

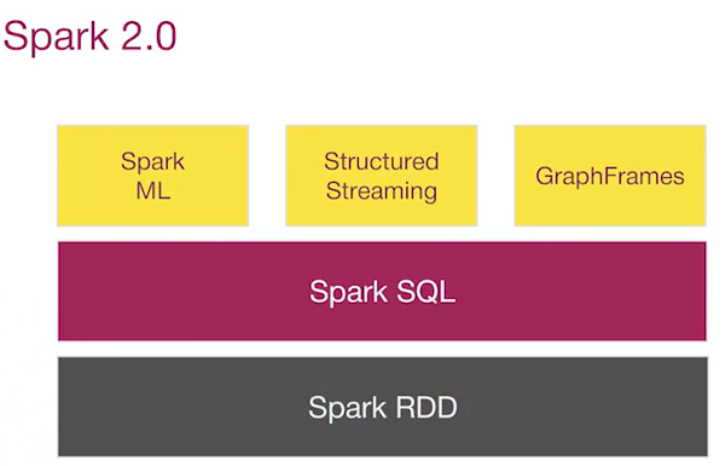
The most powerful of information is structured. And there is a special language for structured data processing, SQL. That stands for Structured Query Language. HIVE allows to process big data in a common SQL language. This is the main reason why HIVE has become one of the most popular data processing tools. All the previous tools like MapReduce operate with raw big data. And it was a real pain in the neck not to say worse to join different data sets or write complex pipelines. HIVE allowed to simplify data processing using MapReduce. Now HIVE users don't need to parse data on disk as it is already parsed and the structure saved in a separate database. They also don't have to write joints at MapReduce each time they need to merge data sets. HIVE could use SQL join clause instead. Nevertheless, MapReduce framework is quite slow. It stores intermediate and final results on the disk between different MapReduce steps. If I could execute hive queries on Spark, Spark works ten times faster during a general data processing and 100 times faster if it can cache intermediate results in memory.



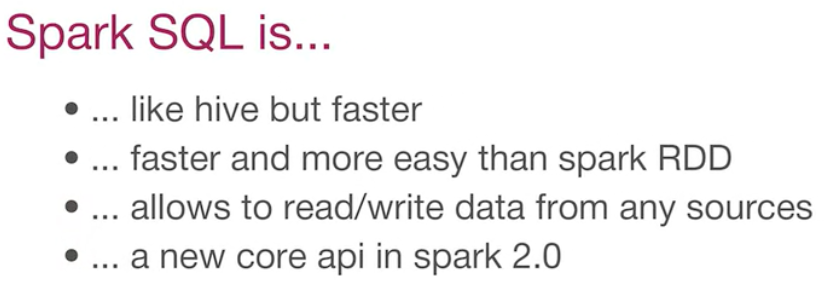
Structured data processing on Spark has advantages of transforming data using original algorithm.

* Firstly, you don't have to parse data every time you need to work with it. Data structure simplifies your code and makes it more readable.
* Secondly, your code is optimized before execution. For example, optimizer could choose between a maps side join, a reduce sidejoin for you and make your code work in a more efficient way.
* Thirdly, you can write your Spark processing on a typical SQL and make your life easy not worrying about RDD syntax.
* And finally, when you SQL query is parsed and executed, all data processing steps will be done in Java without any overhead on parse and code execution. It also makes SQL on Spark much faster than RDD API.
* One more Spark SQL benefit is that you can read and write your data from another structured data sources like JSON files or any external databases. All additional steps with data input and export become optional. Everything can be done inside your Spark job. All these things are becoming real for you when you use Spark SQL and DataFrame framework.





All these things are becoming real for you when you use Spark SQL and DataFrame framework. In Spark 1.0, data frame API was one of the top level components for Spark API that worked on top of Spark RDD. All the same, in Spark 2.0 Spark SQL turned to be a main API. And Spark RDD now is just an internal implementation of it. And all top level libraries are being re-written to work on data frames.



Here is a quick summary. You have learned that Spark SQL is like HIVE but faster. Being similar to RDD but easier and faster and allows you to integrate with different data sources. And it's a main API on Spark 2.0.

#### **3.1.2 What is Pandas DataFrame and how to create it**

In this video you will get acquainted with the key elements of Spark SQL which is called a DataFrame. What is it? Why does it need a schema? How is it related to RDD, how to create them, and how to work with them?

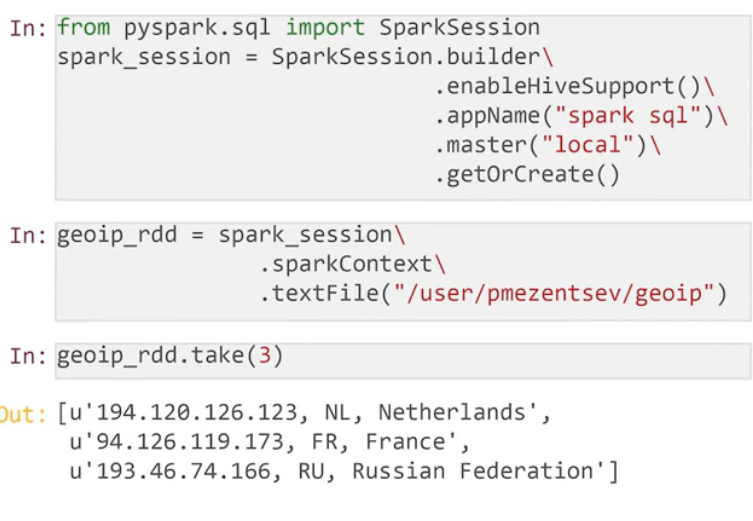
The core element that you will work with in Spark SQL is called a DataFrame.

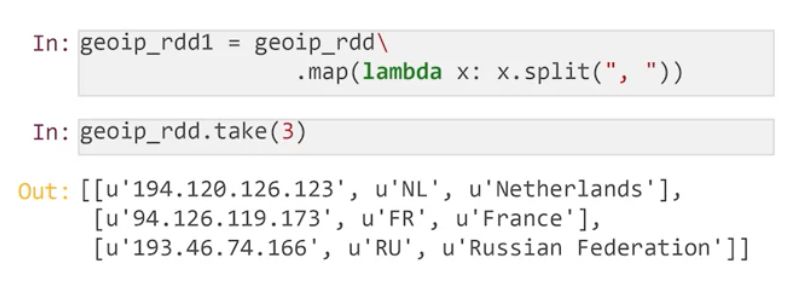
But, what is it? When you are working with structured data, it can be represented as set of tables where each row is an individual record. For example, bank transaction or a logged internet event and each column is a property of that record. Who is it the web page, when it was created, what URL it has, etc.

Tables in Spark SQL are represented by DataFrame objects which allow you to access their data and processes.It's part of DataFrame, it's a distributed collection where I could assign them distributed data sets. Actually DataFrame is built on top of RDD. Each table's row is stored as a separate element of RDD.

But what about columns? In which way does DataFrame works with them?

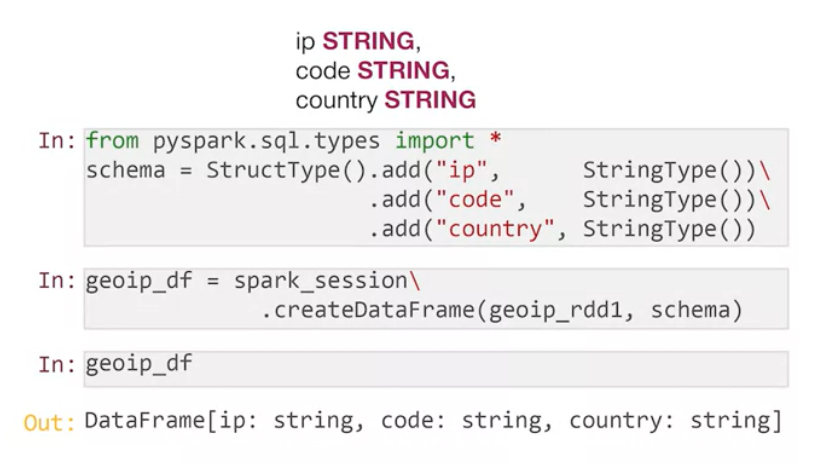
DataFrame also gives additional property called schema that describes the columns of its data. Let's see how this works.

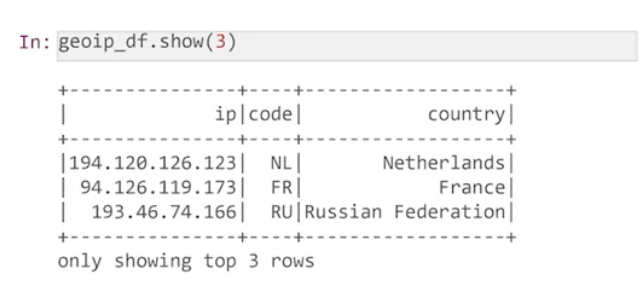




That's how you create a SparkSession object just the way you have done it before. You are starting few RDD API. Now we are reading textual data set from HDFS. Let's see how it actually works.

You have a mapping table in this data set, where each ID is associated with country and its country code. Fields are separated by commas and spaces. You can use such a table to determine user country by its ID. You are able to split its rows by commas and get list of fields, IP, country code, and country name from each report.





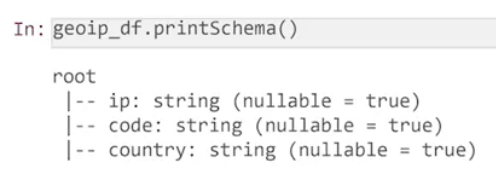
Now you will convert RDD into DataFrame explicitly. To do this, you should define a scheme of dataframe that which column it has. All IP code and country fields are textual and our schema will look like this.

You'll create the object of structure type for the schema and add fields with the names and types for it. All columns store textual data so the type of each column will be string type. Now you can create dataframe from RDD and Schema.

Let's check if you have exactly the data frame you need. We're going to put a presentation of the new object, show us data frame with fields, IP, code and country.



RDD property gives us access to internal RDD where actual data is stored. As you see, data in DataFrame is stored as a set of row objects. Each of them contains the name of the column and its area.



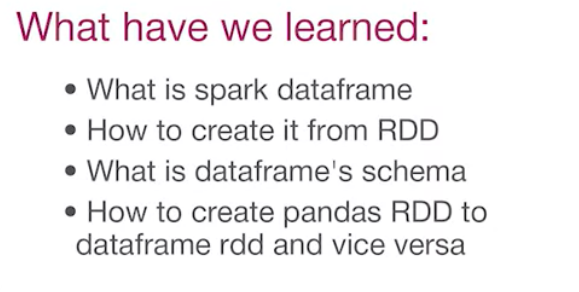
Also, you can check the schema of our DataFrame by print schema message. It will show us a list of columns with their types. NULL property means that field will contain null if the word is missed. Here you can see three columns as planned.



Just call toPandas method of Spark DataFrame and you will receive Pandas DataFrame object. But be careful. The whole data that is stored in your data frame will be collected into your driver. And if you try to convert one terabyte dataset from Spark DataFrame to Pandas DataFrame, your program will run out of memory and crash.

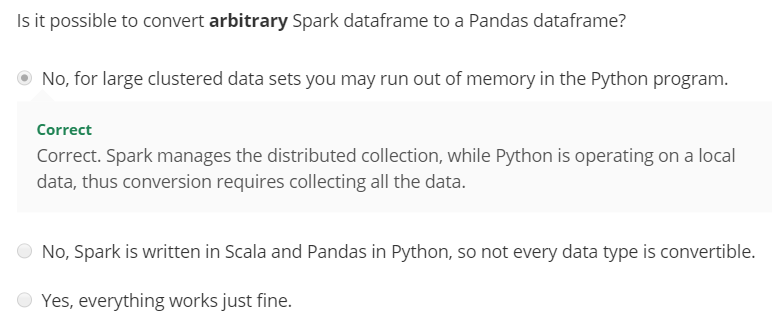
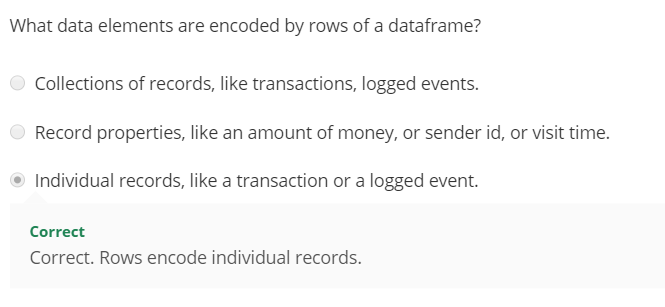


If you need to convert Pandas DataFrame to the Spark one, you can call create dataframe method of Spark session and pass your Panda's object as an input parameter. Your local dataframe will be distributed over Spark executors and you will be able to process it by Spark.

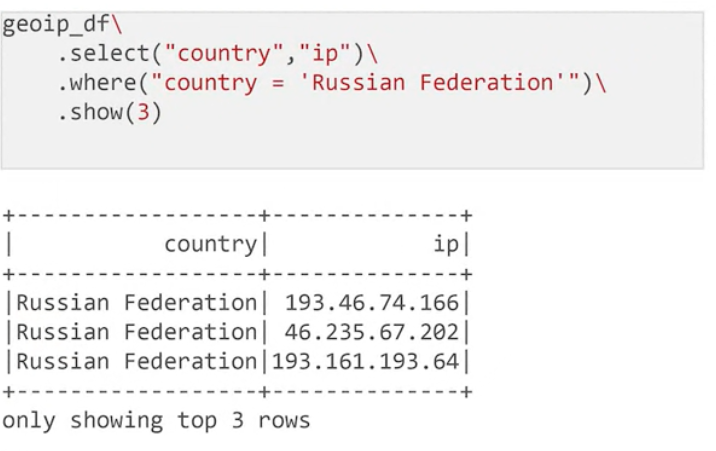


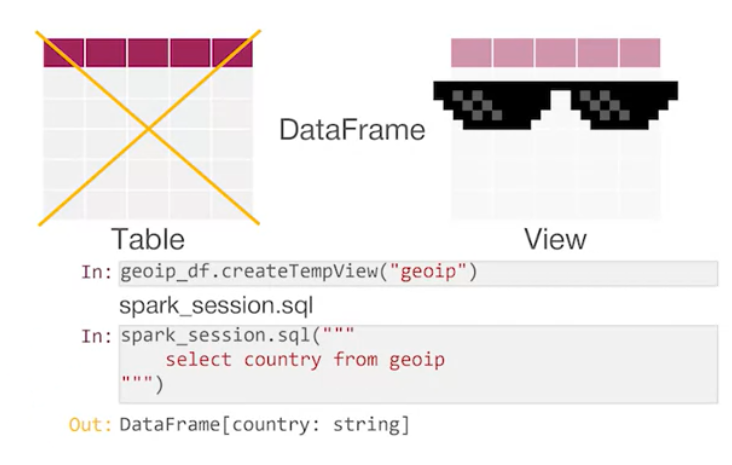
In this video, you have learned what the Spark RDD and how its schema looks like and now you can create it from a simple RDD object and from Pandas DataFrame.

##### **Quiz**



#### **3.1.3 How to process a DataFrame as SQL**





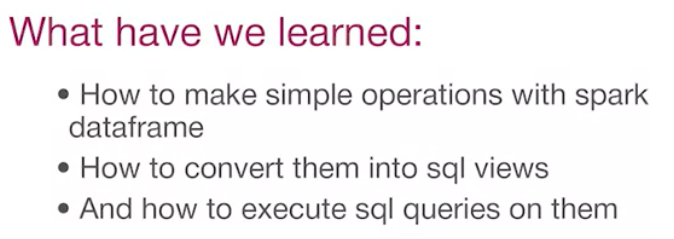


More precise analog of dataframe in SQL is a view. Because both dataframes and views don't give data in fact, but calculate it on request. As you can process these dataframes by SQL queries. Let's see how you can do it.

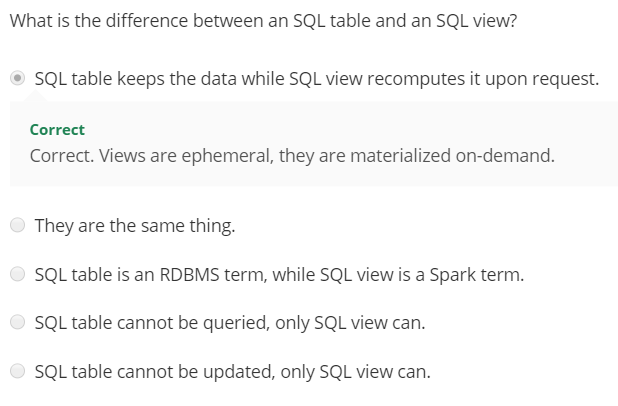
Firstly, you should create a SQL view associated with dataframe. You can do it by the createTempView function. This view is called temporary, because it exists only during the spark session, and will disappear right after it's finished.

And now, you can use the new view in our sql queries. It can be done by spark session SQL method. Just type your query and execute it.Each result of this sql query in Spark SQL is a dataframe object. And you can deal with it as with any typical dataframe. And every dataframe can be converted into SQL view and used in following queries.

##### **Summary**



##### **Quiz**





#### **3.1.4 Working with Hive**

We will talk about how to work with HIVE and Spark SQL. If you are able to connect Spark SQL to HIVE, and use its tables in our database process, then our capabilities will grow significantly. To connect to Hive you should use enableHiveSupport option when you build your Spark session. Here you can see how it's done

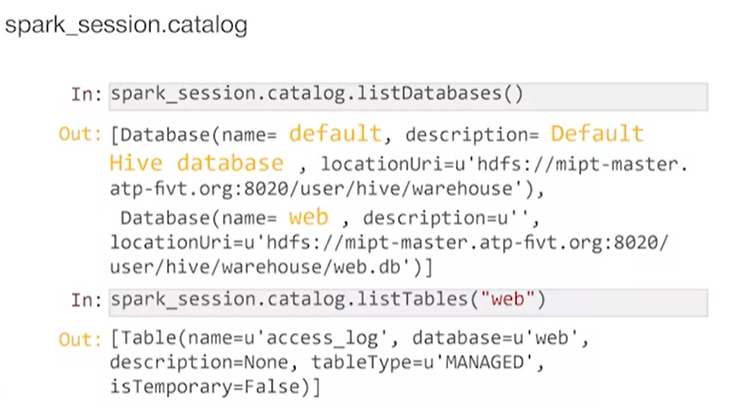


There are two ways to check that Hive tables are available in our Spark session. You can ask Spark to show databases by SQL query. Actually this query is show database.You can list all the tables in any database by show table in query.



The database web contains its access\_log table. You worked a lot with it during the previous week. And there is a way to do this, to use a special catalog field or spark session. The catalog contains methods that work with tables and databases. These databases will return as information about all the registered databases in Hive. And the list table could show you all the tables in one DB. As the output of these functions is not so great as by SQL Query but it contains more information.

For example, you can get the database name and description from its output. One more important difference is that the syntax error in SQL queries will be found on query execution. And in the list database function it will be found when you execute your script, the sooner you will find your error the cheaper it will be to fix it.



Now you are connected to Hive and check that connection is successful. You can run queries on it, as you remember the result of each query is data frame. Let's check that you have access to content of access log table.



And what is more important, you can execute Spark SQL queries on Hive data. The Spark SQL implementation is almost fully compatible with Spark SQL except some tweaky cases.

It means that you can take any Hive query, execute it on Spark SQL and get exactly the same answer.



I have one more question to you, what do you think about Spark SQL query execution? Is it a transformation or an action?

* Actually, it depends on the query that you wrote, for example the query like create table as select stores the result of your query into a new table and return an empty data frame. Such query will be executed as an action and most of other query types are transformations. The Spark SQL optimizer makes a decision about what it will be.
* Now we can download any data set from Hive into Spark SQL.Our next step is starting the process of exporting data frame into the Hive. Now let's return to the geoip data frame. It will be very useful if you keep it together with access\_log. Then you will be able to join them and that reminds the concept of the event. Geographical information is essential in the web log analysis. To do this you should create a temporary view as you did it before.
* If you check the web database again you will find a new geo IP view, the table will have as a temporary flag.That shows that this table is not permanent.

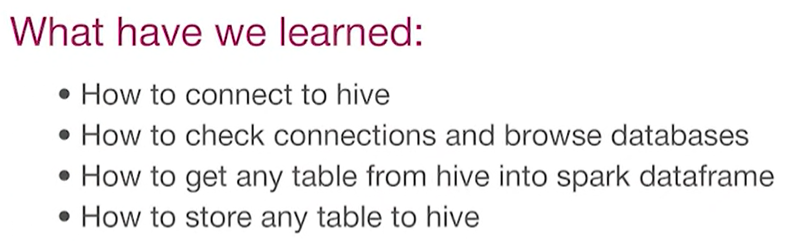


Now you can create a permanent geoip table in web database by create table and select close. If you were to find table software database again, you'll find a new geoip table that is permanent, it has no temporary flag.

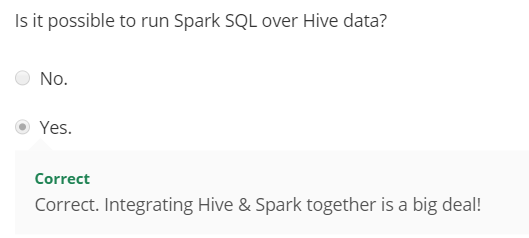


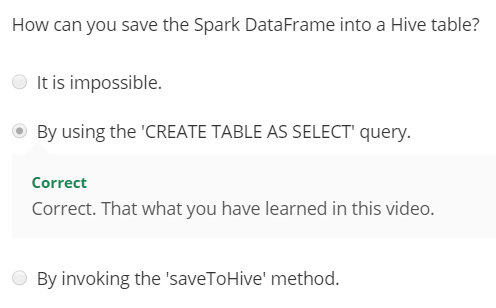
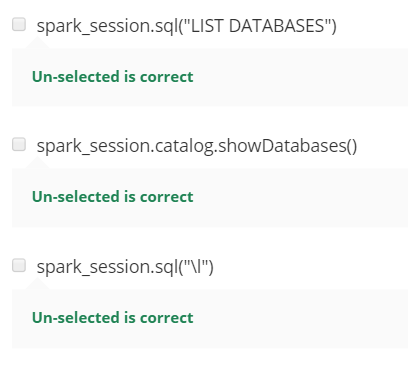
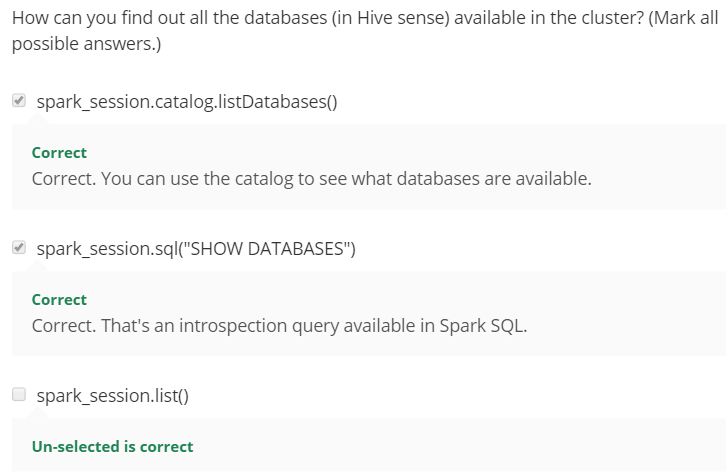
Finally, you can select a few rows from this table and ensure that it contains our geographical data.

##### Summary:



##### **Quiz**





#### **3.1.5 Reading and Writing Files**

In previous video, you learned how to write queries on hive tables. You can use SQL queries to read data from a table, and save it into new one. Now, you will extend your knowledge by learning more ways to read and write data from different sources. At this point, you can save the geoIP data frame into the Hive by SQL query. To do this, you should create a view from our data frame, and execute as a create table a select query on it. After that, you will have a new temporary view with a new table in the web database. But this table creation is verified only when this query is executed, and that will be sad if your program fails at the end of the data process.





You also can save data in the Hive by the spark API method. And in case of a syntax error, your problem will fail at the very beginning, and this will save you a lot of time and nerves. Data frames have a special property write, to save data into any place. You can save data into Hive table by saveAsTable as table method. Let's try this. Unfortunately, the saveAsTable method fails. Let's check why. The table geoIP already exists. You created it by the SQL query earlier.



Let's save our data frame into a new table. Now this works, but what should you do, if for example, you have a new results that are more precise than the previous, and you want to keep them. The default mode is error, and spark slow an exception each time the data already exist in the source. If you want to clean up the previous results, and rewrite a new one, you can use that rewrite mode, and that mode will add the data frame as you wrote in the existing data.



The data frame wide property allows a stored data frames not only to Hive, but the files as well. For example, we can save our table or data in the file by save common. Spark creates geoIP out folder in our house directory, and writes the data there. I suggest you to look what is on the inside. The first three rows taken from the files written as some kind of mess, but you can find elements of data there. Here's an IP address for example. It means that data is stored in a binary format. By default spark works with binary parquet files, which are designed to high performance we can write in.



If you want to store the data in a more human readable form, you can save it in CSV format for example. The data will be organized as a text. The columns in each row will be separated by commas.

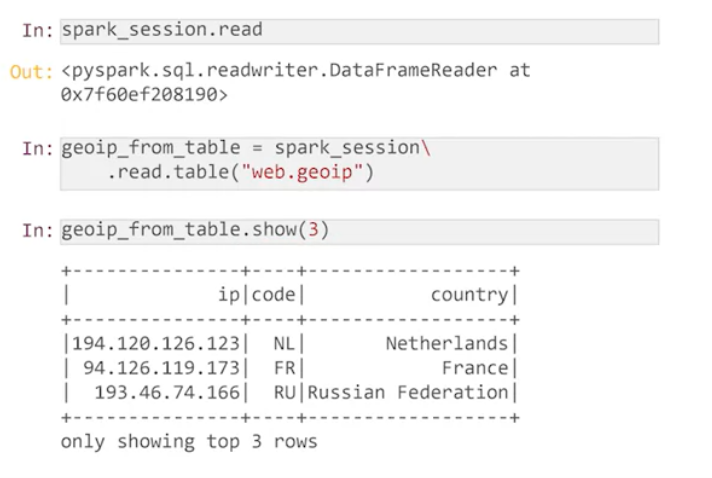


And a popular textual data format is JSON. You can store our data as JSON strings, and each row will contain columns, names, and radius. This overhead help us parsing the data is a table in advance.

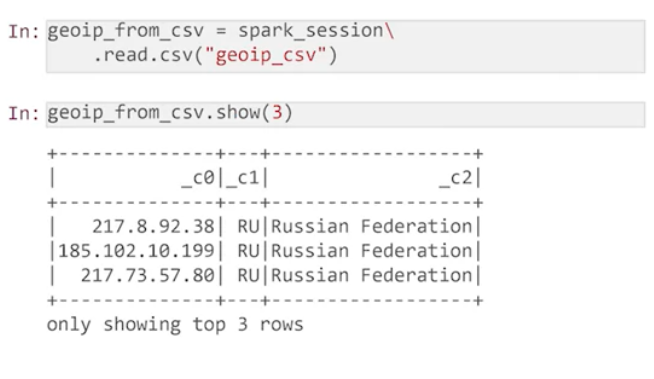


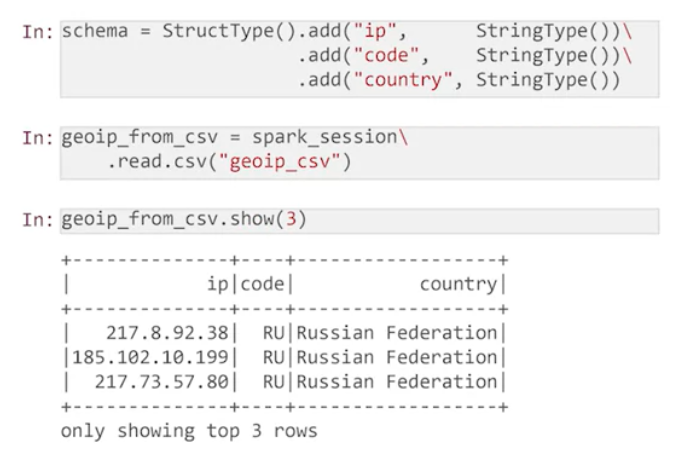
Also, each data format has its explicit function to save. Parquet saves into parquet files, CSV saves into a CSV, JSON saves into JSON. You can choose which one is more convenient for you.

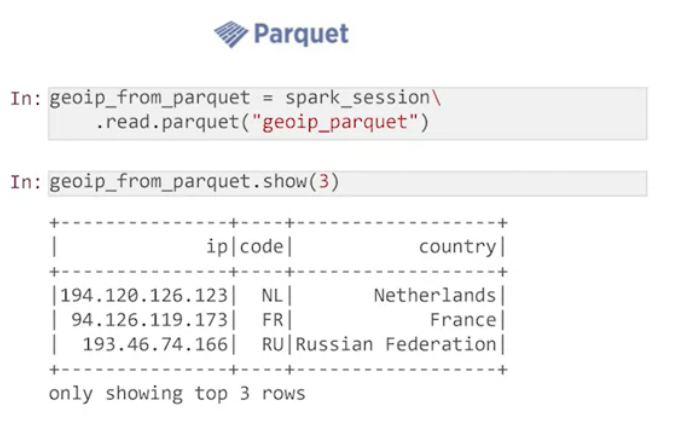
Spark SQL allows to read data from folders and tables by Spark session read property. The spark session read table will create a data frame from the whole table that was stored in a disk. And now you check its first rows.

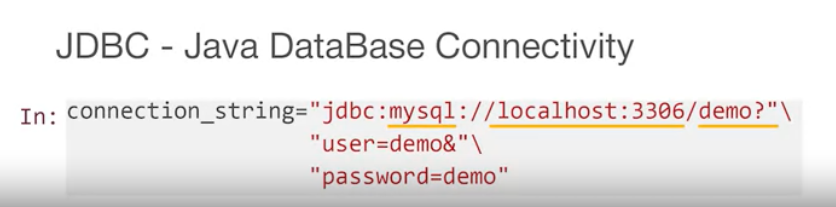
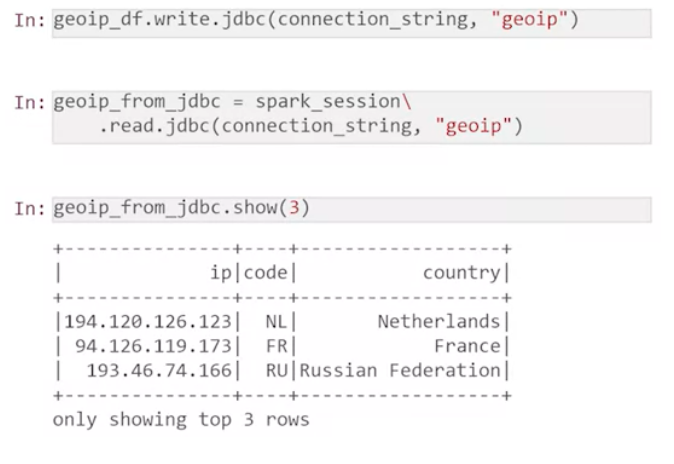
Towards a folder with JSON object, you can use that with JSON method. The columns read from JSON file will be permuted, because the columns in JSON don't have any order.

When you read the data from the CSV file by the read CSV common, it returns the right order of columns, but without the names of columns.

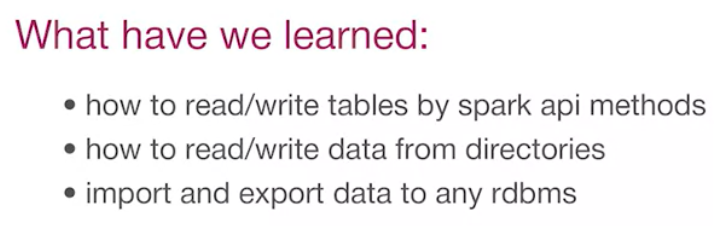
You should provide a schema of the data frame as you did it when we creating the data frame manually. Now the columns are correct,

and you can read data from the parquet file as well. There are no issues with the names of the columns and order in them, because each of the saved parquet files, stored in its own schema.

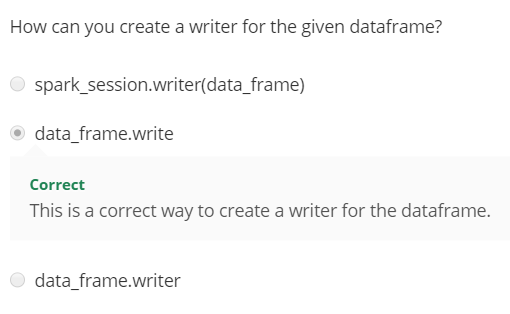
As you remember, each table in the Hive is saved in HDFS as a separate directory. So, save data in table and save data in file are pretty similar.

Now, I will show you saving data into any external database.Each relational database allows to read and write the data into it, by special JDBC protocol. You should just download the jar file with a client library for this database and set a proper connection string. The connection string contains a type of database, its web address, the database name, the username, and the password. Now you can write your data frame into the external database by JDBC, and read its data from JDBC as a new data frame. It is very useful at the end of data processing, when you have created a report, or machine learning prediction, and should make it available as a business system in your company.

##### **Summary**



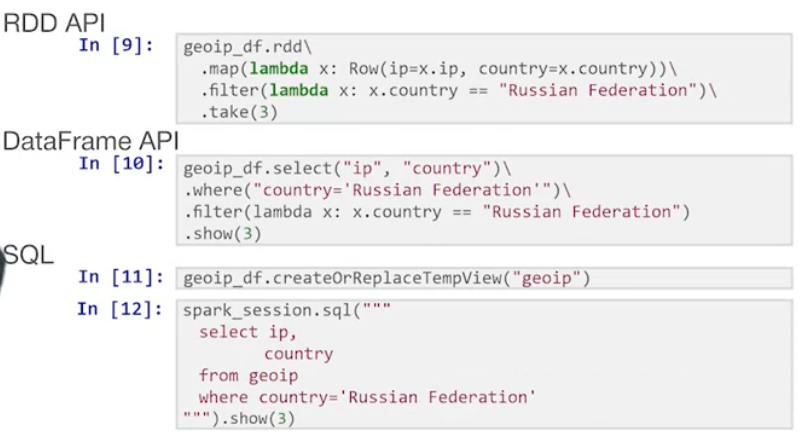
##### **QUIZ**



### **3.2 Complex Operations with DataFrames**

#### **3.2.1 RDD vs DF vs SQL**

By this moment you have already got familiar with three ways of parsing data on the Spark. Take our usual geographic data set. You can select columns and rows in it, using the RDD API, DataFrame API and SQL.

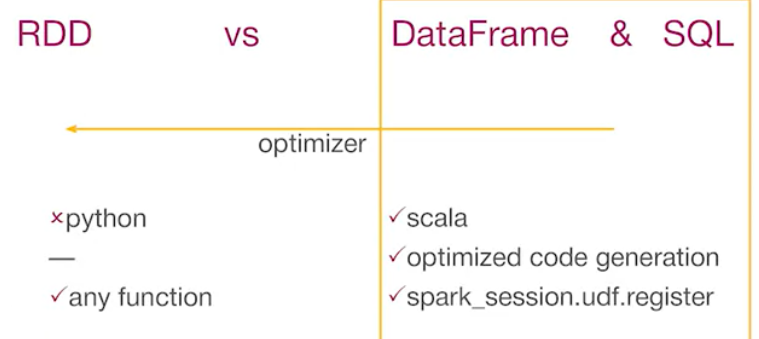


So which one should you choose?

As you already know, DataFrame and SQL are converted by the optimizer into the segments of RDD transformations. At first sight RDD and DataFrame should be executed in the same way.

* However, you write RDD transformation on the Python. It will be executed on workers in a separate process that runs more slowly and has an overhead to receive and return data.
* And the DataFrame optimizer generates a code in the Scala language which runs insides the Java virtual machine without any overhead. As a result, calculations are two, three times faster and as an advantage, the optimizer can choose the optimal way to execute the code. To do filtering before or after segregation, to make a joint user RDD joint or way of broadcast of a smaller table to nodes. In some cases, the processing speed can increase at times.

Some skeptics will object. What if we want to process data in some tricky way? After all, in RDD transformation, we can substitute any function and in Spark SQL we can’t.To do this there is a special command spark\_session.udf.register which makes any of your function available in your SQL code.

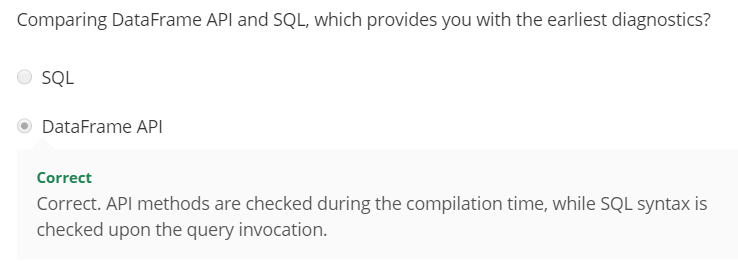
Good, I think I have convinced you to prefer DataFrame to RDD. At this point let's switch on the comparing data frame API, to SQL. Here are two pieces of the code that do the same thing. One is written on the DataFrame API, the second in the SQL query. The code will be converted by optimizer SQL equally and the performance will be the same.

However, if you look closely, there are errors in the code.Here they are. And if you run this code, the example on the left will not start at all due to a compilation error. And the example on the right, will be dropped at the time of the query call. The principle of good code development says, the sooner you will find an error, the easier and cheaper it will be to fix it. In the case of the DataFrame API, we will find it immediately and runs the correct that code again. And in the case of the SQL command, you will see the error only once the query is called. Imagine that this create table as select query program works for two hours and saves the results on the disk with the help of this query. Your results will be lost and the mistake in it will cost you much more.

I can give you an advice. If you already have the working code on the hive, you can just use copy/paste and work with it. If you need to write codes from scratch, use a DataFrame API. Checking the code compilation will save you a lot of time and nerves.



##### **QUIZ**



#### **3.2.2 Projection and Filtering**

#### **3.2.3 Functions**

#### **3.2.4 Aggregates**

#### **3.2.5 Join**

### **3.3 Integration of DataFrames with Hive, Pandas and ODBC**

## WEEK 4 : Graph Analysis from Big Data Perspective

## WEEK 5 : PageRank and Recent Advances

## WEEK 6 : Spark Internals and Optimization