Working with Spark SQL

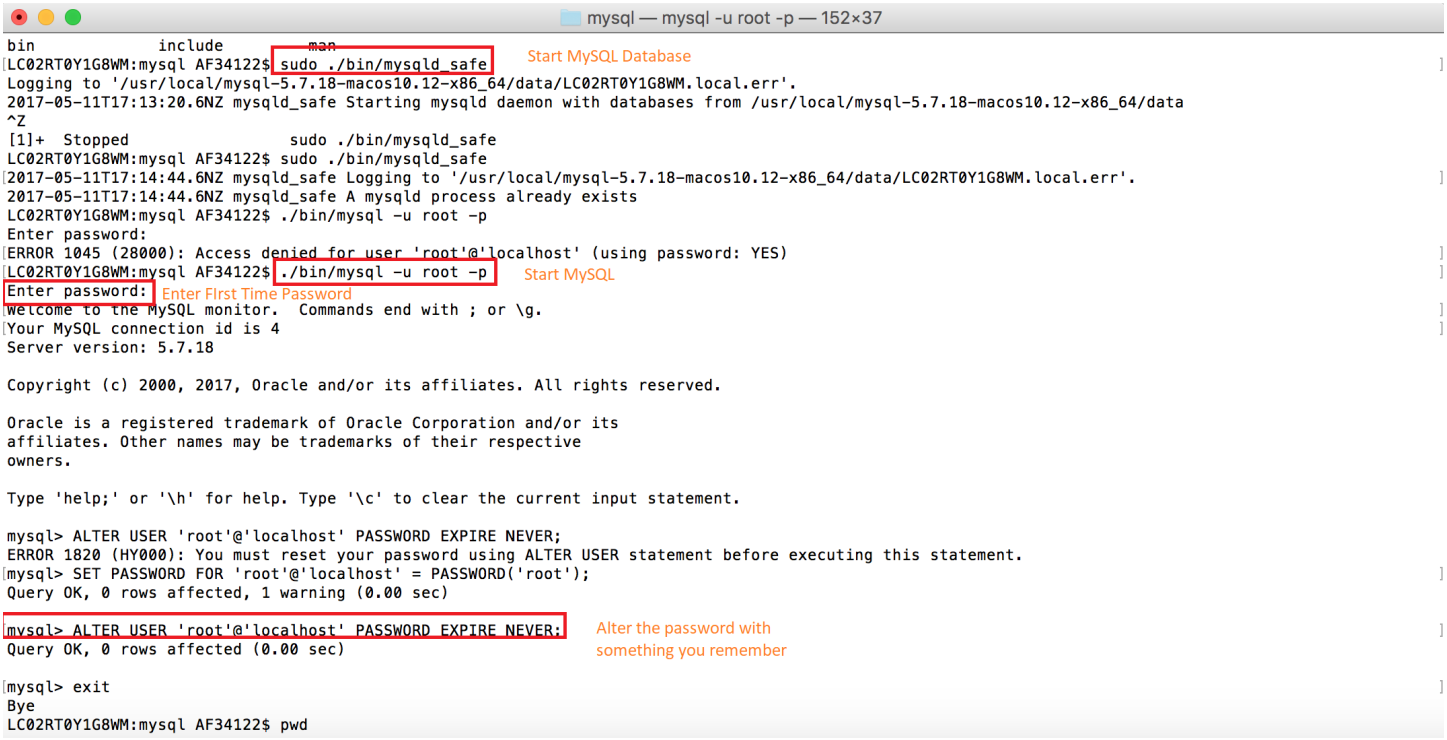
Spark SQL is Spark's module for working with structured data and allows Spark to work with a variety of datasets like CSV, JSON, MySQL tables, etc. It allows user to query structured data inside Spark Programs, using either SQL or Data Frame API.

Data Frames are similar to RDDs and are composed of Row objects, each object accompanied with schema that describes data types of each column. Data Frames can be considered similar to a table in traditional RDBMS.

In order to run this project we will need Spark, Sequel Pro, MySQL Instance and MySQL JDBC driver. I will assume that you have already installed Sequel Pro, MySQL and the JDBC driver for your respective OS.

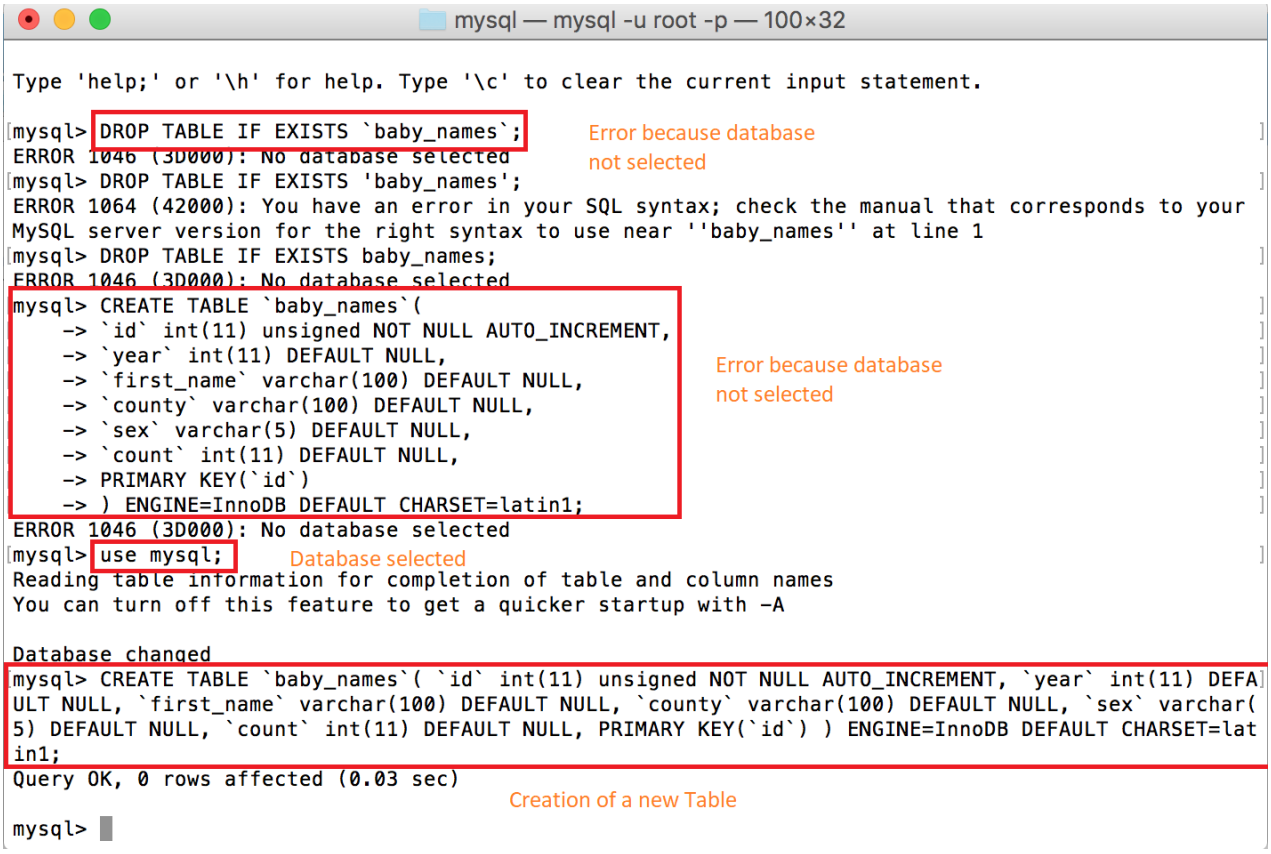
Creation of MySQL and Importing Data:

1. After installing MySQL, open the command line terminal (I am working with Mac OS) and start the MySQL Database server. After the Server starts, run the MySQL with admin privileges. During installation of MySQL, you were given a one-time password for MySQL, use that password to access MySQL. After MySQL starts, alter the password to something you find convenient.

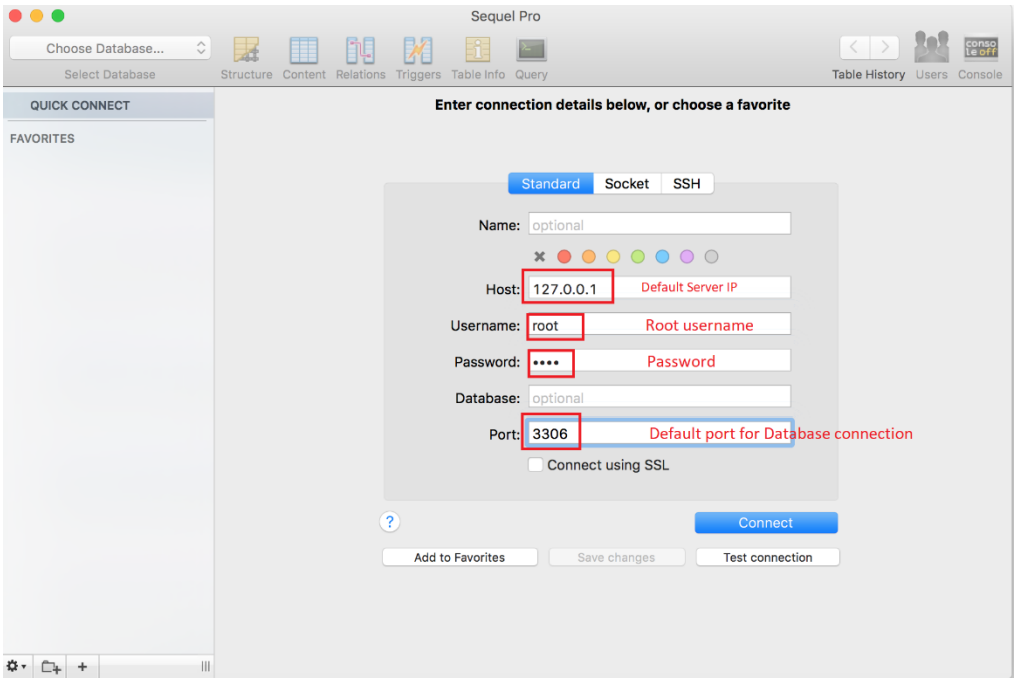



```
bin             include         man
LC02RT0Y1G8WM:mysql AF34122$ sudo ./bin/mysqld_safe
Logging to '/usr/local/mysql-5.7.18-macos10.12-x86_64/data/LC02RT0Y1G8WM.local.err'.
2017-05-11T17:13:20.6NZ mysqld_safe Starting mysqld daemon with databases from /usr/local/mysql-5.7.18-macos10.12-x86_64/data
^Z
[1]+  Stopped                 sudo ./bin/mysqld_safe
LC02RT0Y1G8WM:mysql AF34122$ sudo ./bin/mysqld_safe
2017-05-11T17:14:44.6NZ mysqld_safe Logging to '/usr/local/mysql-5.7.18-macos10.12-x86_64/data/LC02RT0Y1G8WM.local.err'.
2017-05-11T17:14:44.6NZ mysqld_safe A mysqld process already exists
LC02RT0Y1G8WM:mysql AF34122$ ./bin/mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
LC02RT0Y1G8WM:mysql AF34122$ ./bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.18

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'localhost' PASSWORD EXPIRE NEVER;
ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('root');
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> ALTER USER 'root'@'localhost' PASSWORD EXPIRE NEVER;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
LC02RT0Y1G8WM:mysql AF34122$ pwd
```

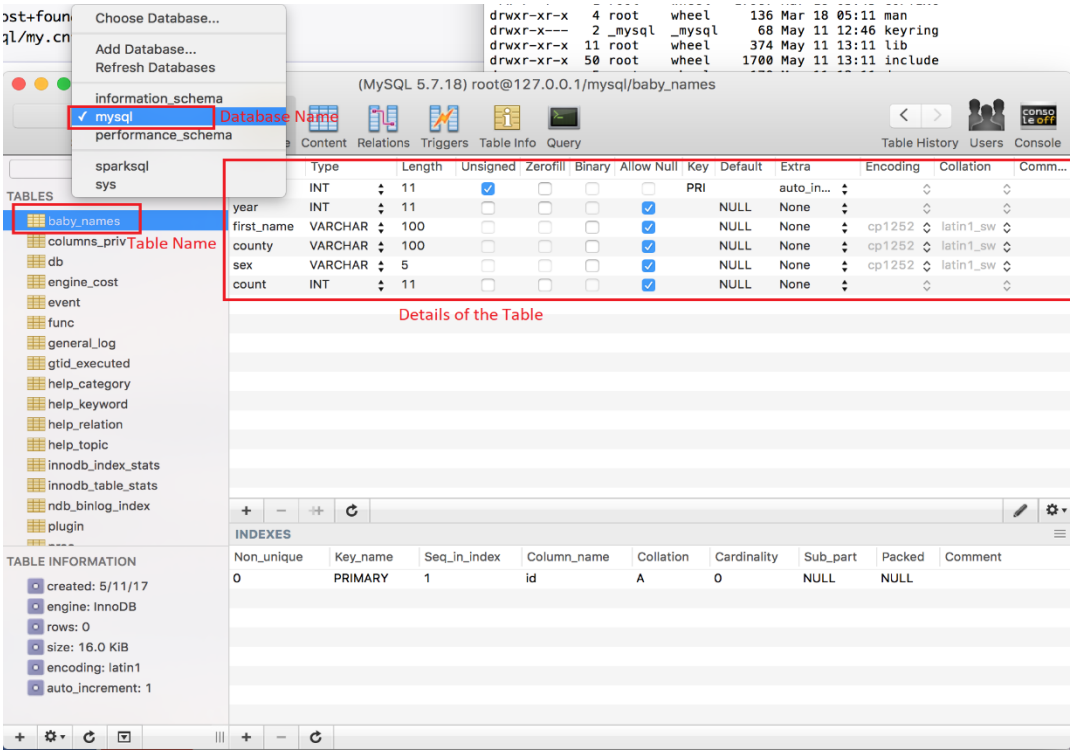
2. After making the changes, login to MySQL again. Before you start making any changes, select or create the database you want to store the tables in. We will be working with the mysql database which is available by default for our project. Once we get access to the database, we create a table which will store the data we will be working on.



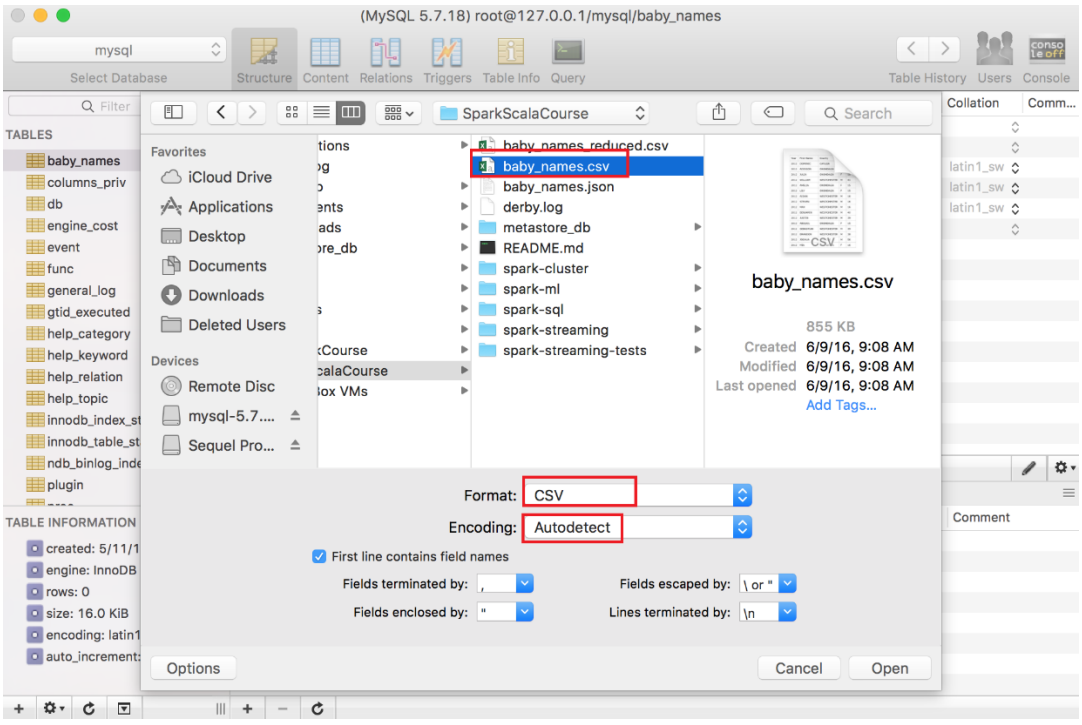
3. Open the Sequel Pro that you have installed and enter the Standard Connection details to connect Sequel Pro to your MySQL server. The info entered would be the same if you had a default configuration for MySQL installation except for username and password. Click “Connect” after entering the details.



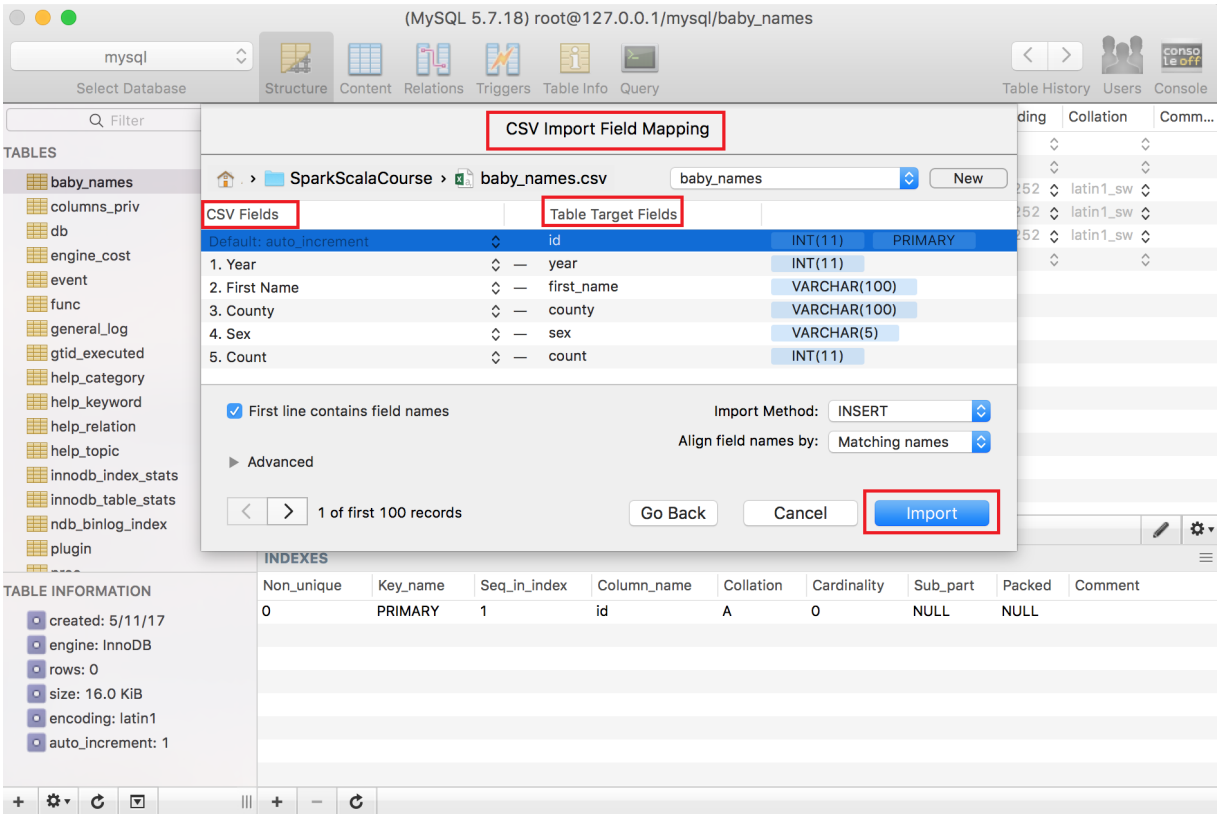
4. After connection has been made, you will see an option on the top-left corner of the window to select the database you want to connect to. Once selecting ‘mysql’, you can choose the table you want to upload your data to. In this case, the table name is ‘baby_names’.



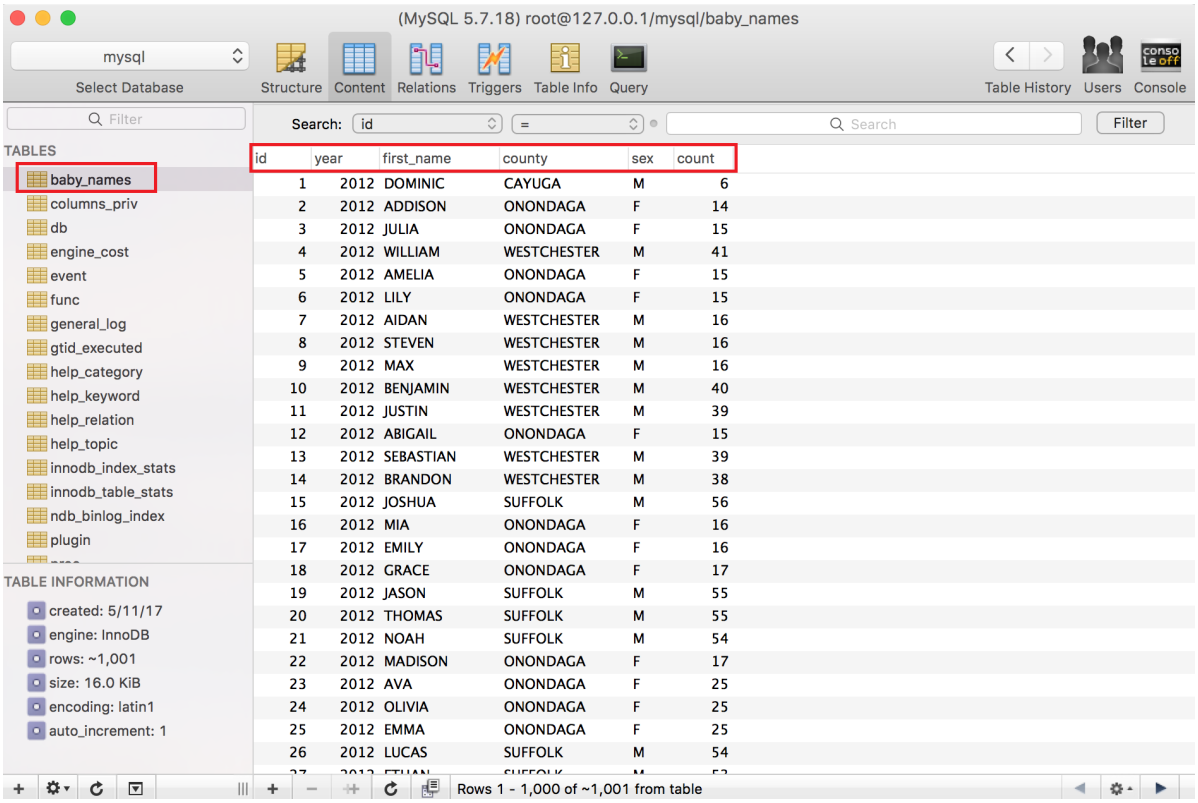
5. After selecting the table name, you have to select the Import option from the File menu, upon which you will have to browse through your file directory and select the file from which you have to import the data into the table. In this case, it is a CSV file. The way in which the file is formatted is important, so if the file you want to import has different delimiters, etc., Then you can manually enter them. Otherwise, Autodetect option for Encoding works just fine.



6. After selecting the “Open” option, you will be taken to another window which will show the CSV fields and the respective Table fields they have been tagged to. Make sure that the correct fields have been linked to the correct columns. Click on “Import” if you are satisfied.



7. Once the import is completed, you will see the data in the selected table in Sequel Pro.



8. You can check if the data has been correctly imported or not in MySQl itself.

mysql — mysql -u root -p — 100×32

```
| 35190 | 2007 | ABIGAIL     | ST LAWRENCE | F    |     7 |
| 35191 | 2007 | ABIGAIL     | DUTCHESS    | F    |    12 |
| 35192 | 2007 | ABIGAIL     | QUEENS      | F    |    64 |
| 35193 | 2007 | ABIGAIL     | ERIE        | F    |    28 |
| 35194 | 2007 | ABIGAIL     | ALBANY      | F    |    11 |
| 35195 | 2007 | ABIGAIL     | WAYNE       | F    |     7 |
| 35196 | 2007 | ABIGAIL     | RENSSELAER  | F    |    14 |
| 35197 | 2007 | ABIGAIL     | NIAGARA     | F    |    11 |
| 35198 | 2007 | ABIGAIL     | NASSAU      | F    |    52 |
| 35199 | 2007 | ABIGAIL     | ORANGE      | F    |    19 |
| 35200 | 2007 | ABIGAIL     | KINGS       | F    |    81 |
| 35201 | 2007 | ABIGAIL     | BROOME      | F    |    12 |
| 35202 | 2007 | ABIGAIL     | RICHMOND    | F    |    15 |
| 35203 | 2007 | ABIGAIL     | JEFFERSON   | F    |     9 |
| 35204 | 2007 | ABIGAIL     | TIOGA       | F    |     5 |
| 35205 | 2007 | ABIGAIL     | NEW YORK    | F    |    46 |
| 35206 | 2007 | ABIGAIL     | ULSTER      | F    |    13 |
| 35207 | 2007 | ABDULLAH    | KINGS       | M    |    11 |
| 35208 | 2007 | ZACHARY     | ST LAWRENCE | M    |     5 |
| 35209 | 2007 | ZACHARY     | KINGS       | M    |    35 |
| 35210 | 2007 | ZACHARY     | SARATOGA    | M    |     6 |
| 35211 | 2007 | ZACHARY     | ROCKLAND    | M    |     8 |
| 35212 | 2007 | ZACHARY     | ONONDAGA    | M    |    18 |
| 35213 | 2007 | ZACHARY     | OSWEGO      | M    |     5 |
| 35214 | 2007 | ZACHARY     | ORANGE      | M    |    12 |
| 35215 | 2007 | ZACHARY     | CHENANGO    | M    |     5 |
| 35216 | 2007 | YUSUF       | NASSAU      | M    |     5 |
| 35217 | 2007 | YOSEF       | ROCKLAND    | M    |    18 |
+-------+------+-------------+-------------+------+-------+
35217 rows in set (0.03 sec)

mysql>
```

9. Before working with the Table in Spark, we need to start Spark shell with the mysql connecter jar file, which will make the connection between Spark and MySQl possible.

mysql-connector-java-5.1.42 — java ◂ spark-shell --jars mysql-connector-java-5.1.42-bin.jar — 144×38

```
LC02RT0Y1G8WM:~ AF34122$ cd mysql-connector-java-5.1.42/
LC02RT0Y1G8WM:mysql-connector-java-5.1.42 AF34122$ ls
CHANGES                                 build.xml
COPYING                                 docs
README                                  mysql-connector-java-5.1.42-bin.jar
README.txt                              src
LC02RT0Y1G8WM:mysql-connector-java-5.1.42 AF34122$ cd ..
LC02RT0Y1G8WM:~ AF34122$ ls
Applications                    Movies                          SparkScalaCourse
Desktop                         Music                           VirtualBox VMs
Documents                       Pictures                        derby.log
Downloads                       Public                          metastore_db
Library                         PysparkCourse                   mysql-connector-java-5.1.42
LC02RT0Y1G8WM:~ AF34122$ cd mysql-connector-java-5.1.42/
LC02RT0Y1G8WM:mysql-connector-java-5.1.42 AF34122$ spark-shell --jars mysql-connector-java-5.1.42-bin.jar
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/05/11 14:42:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/05/11 14:42:28 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording t
he schema version 1.2.0
17/05/11 14:42:28 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
17/05/11 14:42:30 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
Spark context Web UI available at http://30.42.221.237:4040
Spark context available as 'sc' (master = local[*], app id = local-1494528145470).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.1
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

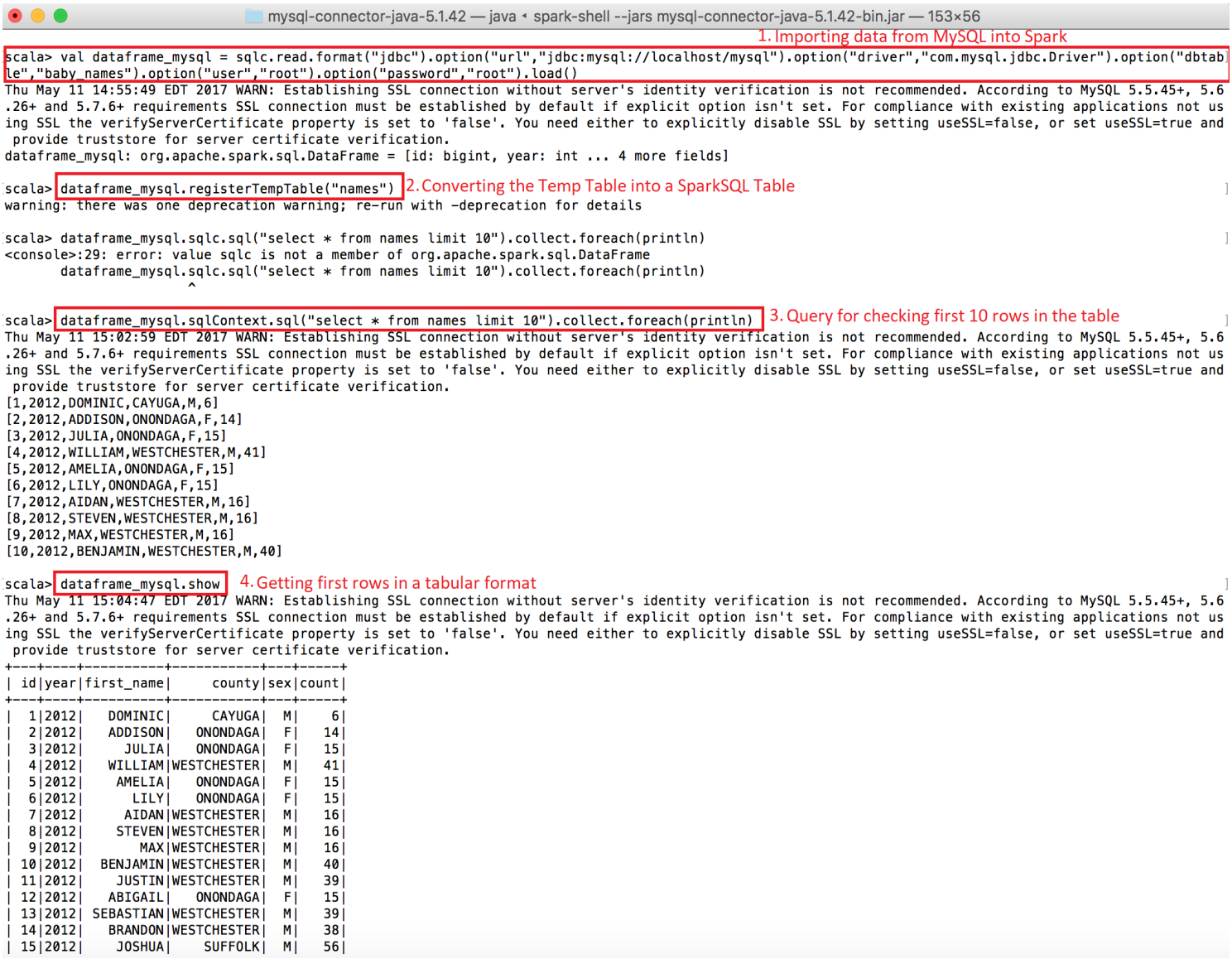
Loading Table data into Spark Dataframe and performing basic queries:

1. After loading data into MySQl table, we will use the MySQL JDBC to import the same data from MySQL into Sparks dataframe. This will be the Temp Table for storing data before it can be converted into Spark SQL table.

2. We next convert the dataframe into a SparkSQL table using the registerTempTable() function followed by the name we want to give to a table. Here, we have named the table as 'names'.

3. We can check a few SQL like queries on the new table within the SparkSQL context. Here, we are checking the first 10 rows of the table.

4. We can still use the old dataframe directly as it is an RDD. We are checking the data of the RDD in tabular format using show().




```
scala> val dataframe_mysql = sqlc.read.format("jdbc").option("url","jdbc:mysql://localhost/mysql").option("driver","com.mysql.jdbc.Driver").option("dbtable","baby_names").option("user","root").option("password","root").load()
Thu May 11 14:55:49 EDT 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
dataframe_mysql: org.apache.spark.sql.DataFrame = [id: bigint, year: int ... 4 more fields]

scala> dataframe_mysql.registerTempTable("names")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> dataframe_mysql.sqlc.sql("select * from names limit 10").collect.foreach(println)
<console>:29: error: value sqlc is not a member of org.apache.spark.sql.DataFrame
       dataframe_mysql.sqlc.sql("select * from names limit 10").collect.foreach(println)
                       ^

scala> dataframe_mysql.sqlContext.sql("select * from names limit 10").collect.foreach(println)
Thu May 11 15:02:59 EDT 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
[1,2012,DOMINIC,CAYUGA,M,6]
[2,2012,ADDISON,ONONDAGA,F,14]
[3,2012,JULIA,ONONDAGA,F,15]
[4,2012,WILLIAM,WESTCHESTER,M,41]
[5,2012,AMELIA,ONONDAGA,F,15]
[6,2012,LILY,ONONDAGA,F,15]
[7,2012,AIDAN,WESTCHESTER,M,16]
[8,2012,STEVEN,WESTCHESTER,M,16]
[9,2012,MAX,WESTCHESTER,M,16]
[10,2012,BENJAMIN,WESTCHESTER,M,40]

scala> dataframe_mysql.show
Thu May 11 15:04:47 EDT 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
+---+----+----------+-----------+---+-----+
| id|year|first_name|     county|sex|count|
+---+----+----------+-----------+---+-----+
|  1|2012|   DOMINIC|     CAYUGA|  M|    6|
|  2|2012|   ADDISON|   ONONDAGA|  F|   14|
|  3|2012|     JULIA|   ONONDAGA|  F|   15|
|  4|2012|   WILLIAM|WESTCHESTER|  M|   41|
|  5|2012|    AMELIA|   ONONDAGA|  F|   15|
|  6|2012|      LILY|   ONONDAGA|  F|   15|
|  7|2012|     AIDAN|WESTCHESTER|  M|   16|
|  8|2012|    STEVEN|WESTCHESTER|  M|   16|
|  9|2012|       MAX|WESTCHESTER|  M|   16|
| 10|2012|  BENJAMIN|WESTCHESTER|  M|   40|
| 11|2012|    JUSTIN|WESTCHESTER|  M|   39|
| 12|2012|   ABIGAIL|   ONONDAGA|  F|   15|
| 13|2012| SEBASTIAN|WESTCHESTER|  M|   39|
| 14|2012|   BRANDON|WESTCHESTER|  M|   38|
| 15|2012|    JOSHUA|    SUFFOLK|  M|   56|
```

Working with CSV files

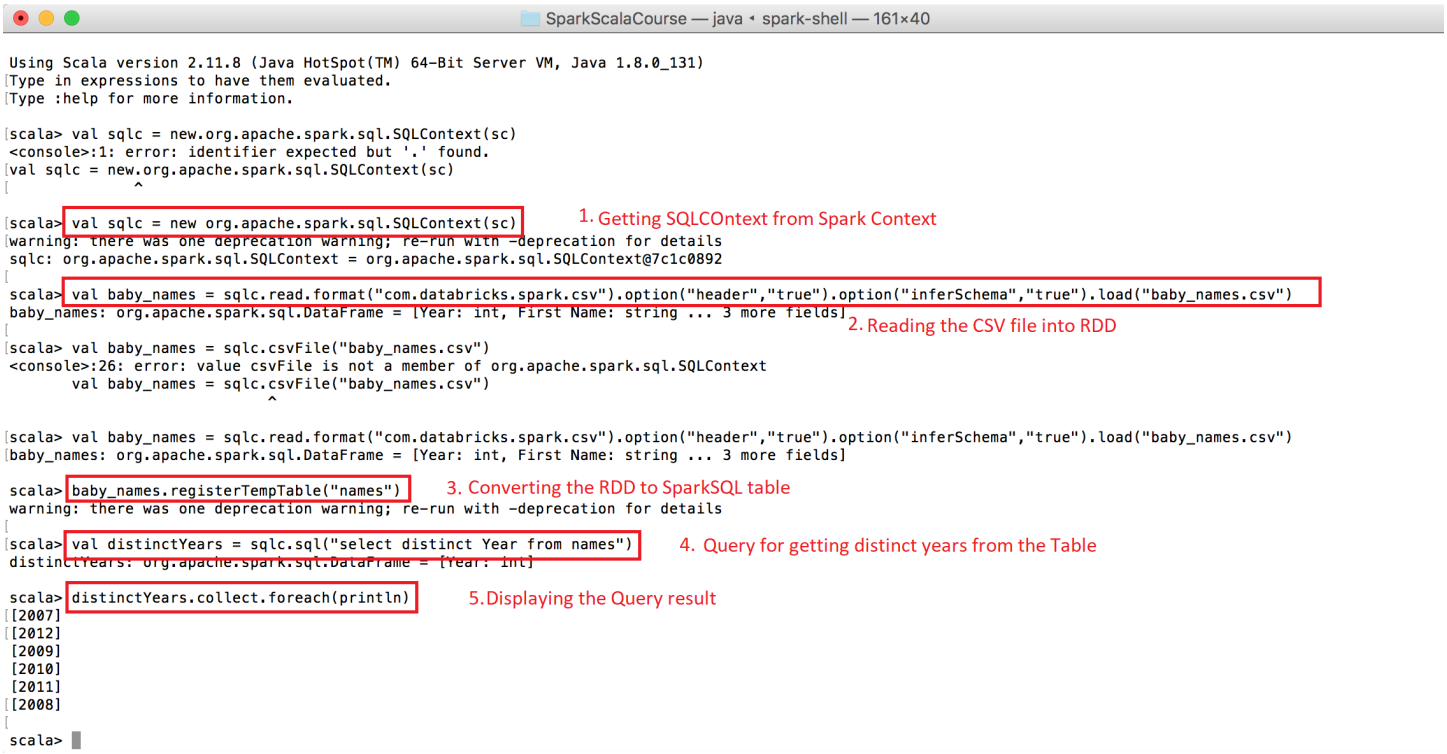
1. We can import a CSV file directly into Spark SQL by first getting SQLContext in Spark.

2. We can then use the jar available from 'com.databricks.spark.csv' with certain parameters to load the CSV in a Spark RDD.

3. We will then convert the RDD into a SparkSQL table by registering it.

4. We will perform a simple query to get the distinct years in the table into an RDD.

5. We will then display each lines in the RDD.



6. We can use a query to get the distinct First Name by County and then display them in descending order to get the most used names by county.

```
scala> val popular_names = sqlc.sql("select distinct(`First Name`), count(County) as cnt from names group by `First Name` order by cnt desc LIMIT 10")
popular_names: org.apache.spark.sql.DataFrame = [First Name: string, cnt: bigint]

scala> popular_names.collect.foreach(println)
[JACOB,237]
[EMMA,223]
[LOGAN,220]
[OLIVIA,217]
[ISABELLA,209]
[SOPHIA,200]
[NOAH,197]
[ETHAN,195]
[MICHAEL,194]
[MASON,194]
```

7. Another query can just get the most used name in the entire dataset by getting First Name and their Count in descending order.

```
scala> val popular_names = sqlc.sql("select distinct(`First Name`), sum(Count) as cnt from names group by `First Name` order by cnt desc LIMIT 10")
popular_names: org.apache.spark.sql.DataFrame = [First Name: string, cnt: bigint]

scala> popular_names.collect.foreach(println)
[MICHAEL,9187]
[MATTHEW,7891]
[JAYDEN,7807]
[ISABELLA,7782]
[JOSEPH,7609]
[JACOB,7444]
[ANTHONY,7427]
[DANIEL,7313]
[SOPHIA,7274]
[RYAN,7172]
```

Working with JSON files

1. Get the SQLContext from Spark Context and name it sqlc.

2. The queerness with Spark's JSON format is that it is compatible only with JSON files that have 'newline' segmentation. It is not compatible with the 'valid' JSON file, where segmentation with ',' and are encapsulated with '[]', which is the accepted form of JSON form. So, we have to read a valid JSON as a text file, then get the 'key : value' pairs in an RDD.

3. We will then read the RDD as a JSON file and keep it as a Temp Table.

4. We will import this Temp Table into a SparkSQL table.

5. We will perform a query which will get all the names and perform a collect operation on it. This will give us the maximum used names.