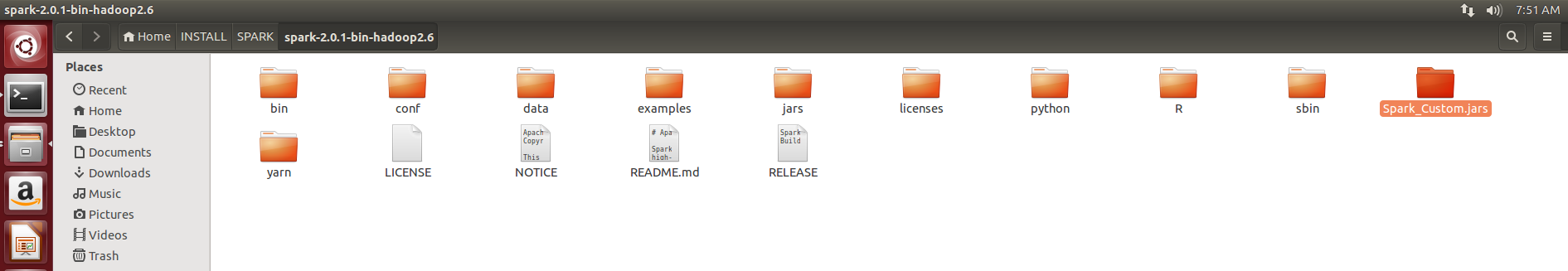
**SPARK SQL – 2.X EXAMPLES WITH “JDBC” INTRACTION**

1. **copy "Spark\_Custom.jars" foder to "$SPARK\_HOME" folder**



1. **Rename "$SPARK\_HOME/conf/spark-defaults.conf.template" file TO "$SPARK\_HOME/conf/spark-defaults.conf"**
2. **Add the below lines to "$SPARK\_HOME/conf/spark-defaults.conf" file**

**spark.driver.memory 5g**

**spark.jars /home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/guava-16.0.1.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/spark-cassandra-connector\_2.11-2.0.0-M3.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/cassandra-driver-core-3.0.2.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/jackson-core-2.5.2.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/jackson-databind-2.5.2.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/jackson-module-scala\_2.11-2.5.2.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/jsr166e-1.1.0.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/mysql-connector-java-5.1.38.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/phoenix-4.7.0-HBase-1.1-client.jar,/home/gopalkrishna/INSTALL/SPARK/spark-2.0.1-bin-hadoop2.6/Spark\_Custom.jars/phoenix-spark-4.7.0-HBase-1.1.jar**

1. **start the `spark-shell`**

**SPARK SQL CODE EXAMPLES**

**scala> import org.apache.spark.\_**

**import org.apache.spark.\_**

**scala> import org.apache.spark.sql.\_**

**import org.apache.spark.sql.\_**

**NOTE: To Connect to RDBMS , we MUST require the below 4 values**

**To connect any RDBMS:**

**1. connection url**

**2. username & password**

**3. driver class name**

**4. client jar**

**FROM MySQL SIDE:**

**mysql> create database gkdb;**

**Query OK, 1 row affected (0.00 sec)**

**mysql> use gkdb;**

**Database changed**

**mysql> INSERT INTO emptab VALUES(100,'RAJ',12000);**

**mysql> INSERT INTO emptab VALUES(101,'RAJU',14000);**

**mysql> INSERT INTO emptab VALUES(102,'RAM',16000);**

**mysql> INSERT INTO emptab VALUES(103,'RAVI',18000);**

**mysql> INSERT INTO emptab VALUES(104,'RAMYA',20000);**

**mysql> INSERT INTO emptab VALUES(105,'RAKESH',22000);**

**mysql> INSERT INTO emptab VALUES(106,'KAVITHA',24000);**

**mysql> INSERT INTO emptab VALUES(107,'KAVYA',26000);**

**scala> val prop = new java.util.Properties**

**prop: java.util.Properties = {}**

**scala> prop.setProperty("driver","com.mysql.jdbc.Driver")**

**res0: Object = null**

**scala> prop.setProperty("user","root")**

**res1: Object = null**

**scala> prop.setProperty("password","root")**

**res2: Object = null**

**scala> val jdbcDF = spark.read.jdbc("jdbc:mysql://localhost:3306/gkdb", "emptab", prop)**

**jdbcDF: org.apache.spark.sql.DataFrame = [id: int, name: string ... 1 more field]**

**scala> jdbcDF.show**

**+---+-------+-----+**

**| id| name| sal|**

**+---+-------+-----+**

**|100| RAJ|12000|**

**|101| RAJU|14000|**

**|102| RAM|16000|**

**|103| RAVI|18000|**

**|104| RAMYA|20000|**

**|105| RAKESH|22000|**

**|106|KAVITHA|24000|**

**|107| KAVYA|26000|**

**+---+-------+-----+**

**scala> val jdbcDF = spark.read.jdbc("jdbc:mysql://localhost:3306/gkdb", "emptab", Array("sal > 24000") , prop)**

**jdbcDF: org.apache.spark.sql.DataFrame = [id: int, name: string ... 1 more field]**

**scala> jdbcDF.show**

**+---+-----+-----+**

**| id| name| sal|**

**+---+-----+-----+**

**|107|KAVYA|26000|**

**+---+-----+-----+**

**TO WRITE THE DATA FRAME DATA IN “JSON” FORMAT**

**scala> jdbcDF.write.json("/home/gopalkrishna/PRAC/emp.json")**

**TO WRITE THE DATA FRAME DATA IN “PARQUET” FORMAT**

**scala> jdbcDF.write.parquet("/home/gopalkrishna/PRAC/emp.parquet")**

**TO WRITE THE DATA FRAME DATA IN “ORC” FORMAT**

**scala> jdbcDF.write.orc("/home/gopalkrishna/PRAC/emp.orc")**

**TO WRITE THE DATA FRAME DATA IN “CSV” FORMAT**

**scala> jdbcDF.write.csv("/home/gopalkrishna/PRAC/emp.csv")**

**scala> jdbcDF.write.mode("append").csv("/home/gopalkrishna/PRAC/emp.csv")**

**scala> jdbcDF.write.mode("overwrite").csv("/home/gopalkrishna/PRAC/emp.csv")**

**TO WRITE THE SAME DATAFRAME DATA TO HIVE TABLES**

**To Have Spark – Hive Integration🡪 copy the $HIVE\_HOME/conf/hive-site.xml and place it in $SPARK\_HOME/conf/**

**scala> jdbcDF.write.saveAsTable("emptabb")**

**NOTE: Go to Hive Shell , and check the above table “emptabb” in default database**

**scala> jdbcDF.write.saveAsTable("gkdb.emptabb")**

**NOTE: Go to Hive Shell , and check the above table “emptabb” in gkdb database**

**TO WRITE THE SAME DATAFRAME DATA TO RDBMS**

**scala> jdbcDF.write.jdbc("jdbc:mysql://localhost:3306/gkdb", "exptab", prop)**

**scala> jdbcDF.write.** **mode("overwrite")jdbc("jdbc:mysql://localhost:3306/gkdb", "exptab", prop)**

**scala> jdbcDF.write.** **mode("append").jdbc("jdbc:mysql://localhost:3306/gkdb", "exptab", prop)**