Spark SQL

Load data from hive:

```
val df = sql("SELECT␣*␣from␣transfers");
```

Load data from csv:

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

val schema = StructType(Array(StructField("src", StringType, true), StructField("dst", StringType, true), StructField("amount", IntegerType, true), StructField("date"

val df = spark.read.format("csv").schema(schema).load("/path/to/csv")

// or
val df = spark.read.format("csv").load("/path/to/csv").toDF("src", "dst", "amount", "date")
```

Load data from binary format:

```
val df = spark.read.format("orc").load("/path/to/orc/table")
```

Load data from mysql:

```
val df = spark.read.format("jdbc").option("url", "jdbc:mysql://hadoop-db:3306/test").option("driver", "com.mysql.jdbc.Driver").option("user", "test").option("password
```

Filtering the data:

```
df.where("amount␣<␣5000")
df.filter(\$"amount" < 5000)
df.filter(col("amount") < 5000)
df.filter(r => r.getInt(2) < 5000)
```

Mapping the data:

```
df.map(row => (row.getInt(2) * 2, row.getString(0))).toDF("amount␣times␣2", "src")
```

Sorting:

```
df.orderBy(\$"count".desc)
```

Save data to hive:

```
df.createOrReplaceTempView("tempDfView")
sql("create␣table␣table_name␣as␣select␣*␣from␣tempDfView");
```

Save data to file:

```
df.write.csv("/user/xyz/dir")
df.write.orc("/user/xyz/dir")
```

Perform join:

```
df1.as("df1").join(df2.as("df2"), col("df1.col_name") === col("df2.col_name"), "inner")
```

Tasks

1. perform a word count and save is as table to hive.
2. Count how many incoming transfers were there for each account.
3. Join transfers with owners. And save a file containing name instead of src and dst.