Hadoop: Hands-On:
Spark Basic Commands

Spark Commands:

Start Spark Session:

```
$spark-shell
```

Command 1: Create a DataFrame (replace the names with your arguments)

```
val product = List((1,"mobile",50000),(2,"shoes",4500),(3,"TV",70000))
val productDF = product.toDF("pid","product","value") [WITH DBPROPERTIES
(property_name=property_value, ...)]
```

```
scala> val product = List((1,"mobile",50000),(2,"shoes",4500),(3,"TV",70000))
product: List[(Int, String, Int)] = List((1,mobile,50000), (2,shoes,4500), (3,TV,70000))

scala> val productDF = product.toDF("pid","product","value")
productDF: org.apache.spark.sql.DataFrame = [pid: int, product: string ... 1 more field]
```

Command 2: Create a DataFrame with JSON (replace the names with your arguments)

```
val df = spark.read.json("/student1.json")
df.show()
```

```
scala> val df = spark.read.json("/student1.json")
2018-12-17 14:44:56 WARN  ObjectStore:568 - Failed to get database global_temp, returning NoSuchObjectException
df: org.apache.spark.sql.DataFrame = [age: bigint, name: string]

scala> df.show()
+----+------+
| age|  name|
+----+------+
|null|   Sam|
|  17|  Mick|
|  18|Jennet|
|  19|Serena|
+----+------+
```


Command(s) 3: Basic Spark Commands (Print Schema, Select, Show, Filter, and Group By)

```
import spark.implicits._
df.printSchema()
df.select("name").show()
df.select($"name", $"age" + 1).show()
df.filter($"age" > 21).show()
df.groupBy("age").count().show()
```

```
scala> df.printSchema()
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)


scala> df.select("name").show()
+-------+
|   name|
+-------+
|    Sam|
|   Mick|
|Jennet|
|Serena|
+-------+
```

```
scala> df.select($"name", $"age" + 1).show()
+------+---------+
|  name|(age + 1)|
+------+---------+
|   Sam|     null|
|  Mick|       18|
|Jennet|       19|
|Serena|       20|
+------+---------+


scala> df.filter($"age" > 21).show()
+---+----+
|age|name|
+---+----+
+---+----+


scala> df.groupBy("age").count().show()
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  17|    1|
|  18|    1|
+----+-----+
```


SPARK SQL

First, import the necessary packages:

```
import org.apache.spark.sql.SparkSession
```

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession
```

Step 1: Using the builder function, create a Spark Session

```
val spark = SparkSession.builder().appName("Spark SQL
basicexample").config("spark.some.config.option", "some-value").getOrCreate()
```

```
scala> val spark = SparkSession.builder().appName("Spark SQL basic example").con
fig("spark.some.config.option", "some-value").getOrCreate()
```

```
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@668
31b14
```

Step 2: Import implicit classes

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

```
scala> import spark.implicits._
import spark.implicits._
```


Step 3: Create a dummy JSON file with the following content (copy and paste)

```
{"age" : "28", "name" : "John"}
 {"age" : "36", "name" : "Andrew"}
 {"age" : "22", "name" : "Clarke"}
{"age" : "42", "name" : "Kevin"}
 {"age" : "51", "name" : "Richard"}
```

Open

employee.json
~/

```
{"age" : "28", "name" : "John"}
{"age" : "36", "name" : "Andrew"}
{"age" : "22", "name" : "Clarke"}
{"age" : "42", "name" : "Kevin"}
{"age" : "51", "name" : "Richard"}
```

Step 4: Create a DataFrame using a JSON file

```
val df = spark.read.json("employee.json")
df.show()
```

```
scala> val df = spark.read.json("employee.json")
df: org.apache.spark.sql.DataFrame = [age: bigint, name: string]

scala> df.show()
+---+-------+
|age|   name|
+---+-------+
| 28|   John|
| 36| Andrew|
| 22| Clarke|
| 42|  Kevin|
| 51|Richard|
+---+-------+
```


Step 5: Increment all ages by 2 and show the output

```
df.select($"name", $"age" + 2).show()
```

```
scala> df.select($"name", $"age" + 2).show()
+-------+---------+
|   name|(age + 2)|
+-------+---------+
|   John|       30|
| Andrew|       38|
| Clarke|       24|
|  Kevin|       44|
|Richard|       53|
+-------+---------+
```

Step 6: Filter all the employees above age 30 and display the result

```
df.filter($"age" > 30).show()
```

```
scala> df.filter($"age" > 30).show()
+---+-------+
|age|   name|
+---+-------+
| 36| Andrew|
| 42|  Kevin|
| 51|Richard|
+---+-------+
```


Step 7: Count the number of employees with the same age, using the 'groupBy' function

```
df.groupBy("age").count().show()
```

```
scala> df.groupBy("age").count().show()
+---+-----+
|age|count|
+---+-----+
| 22|    1|
| 51|    1|
| 28|    1|
| 36|    1|
| 42|    1|
+---+-----+
```

Step 8: Create a temporary view 'employee' of the df DataFrame and perform a 'select' operation on it to display the table into 'sqlDF'

```
df.createOrReplaceTempView("employee")
val sqlDF = spark.sql("SELECT * FROM employee")
```

```
scala> df.createOrReplaceTempView("employee")

scala> val sqlDF = spark.sql("SELECT * FROM employee")
sqlDF: org.apache.spark.sql.DataFrame = [age: bigint, name: string]
```


Step 9: Now, display the results of 'sqlDF'

```
sqlDF.show()
```

```
scala> sqlDF.show()
+---+-------+
|age|   name|
+---+-------+
| 28|   John|
| 36| Andrew|
| 22| Clarke|
| 42|  Kevin|
| 51|Richard|
+---+-------+
```

