

Monica Santana – Exercise 5

Scala Commands

```
santanamonica@dscbigdata: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
Type in expressions to have them evaluated.
Type :help for more information.

scala> val df = spark.read.format("csv").option("header", "true").load("/data/grades.csv")
df: org.apache.spark.sql.DataFrame = [last name: string, first name: string ... 7 more fields]

scala> df.createOrReplaceTempView("df")

scala> spark.sql("SHOW TABLES").show()
81898 [main] WARN org.apache.hadoop.hive.conf.HiveConf - HiveConf of name hive
strict.managed.tables does not exist
81899 [main] WARN org.apache.hadoop.hive.conf.HiveConf - HiveConf of name hive
create.table.insert.only does not exist
81950 [main] WARN org.apache.spark.sql.hive.client.HiveClientImpl - Detected H
iveConf hive.execution.engine is 'tez' and will be reset to 'mr' to disable use
less hive logic
=====
[database|tableName|isTemporary]
=====
|      |      |      | |
|      | df |      | true |
|      |      |      |
=====

scala> spark.sql("SELECT * FROM df WHERE Final > 50").show()
=====
[Last name|First name|      SSN|Test1|Test2|Test3|Test4|Final|Grade]
=====
| Airpump| Andrew|223-45-6789| 49| 1| 90| 100| 83| A-|
| Backus|  Jim|143-12-1234| 48| 1| 97| 96| 97| A+|
| Elephant| Tma|456-71-9012| 45| 1| 78| 88| 77| B-|
| Franklin| Benny|234-56-2890| 50| 1| 90| 80| 90| B-|
=====

scala> spark.sql("SELECT * FROM df").show()
=====
[Last name|First name|      SSN|Test1|Test2|Test3|Test4|Final|Grade]
=====
| Alfalfa| Aloysius|123-45-6789| 40| 90| 100| 83| 49| D-|
| Alfred|University|123-12-1234| 41| 97| 96| 97| 48| D+|
| Gerty| Gramma|567-89-0123| 41| 80| 60| 40| 44| C|
| Android| Electric|087-65-4321| 42| 23| 36| 45| 47| B-|
| Bumpkin| Fred|456-78-9012| 43| 78| 88| 77| 45| A-|
| Rubble| Betty|234-56-7890| 44| 90| 80| 90| 46| C+|
| Noshov| Cecil|345-67-8901| 45| 11| -1| 4| 43| F|
| Buff| Biff|632-79-9999| 46| 20| 30| 40| 50| B+|
| Airpump| Andrew|223-45-6789| 49| 1| 90| 100| 83| A-|
| Backus|  Jim|143-12-1234| 48| 1| 97| 96| 97| A+|
| Garatwoe| Art|565-88-0123| 44| 1| 80| 60| 40| D+|
| Dandy|  Jim|087-78-4321| 47| 1| 23| 36| 45| C+|
| Elephant| Tma|456-71-9012| 45| 1| 78| 88| 77| B-|
| Franklin| Benny|234-56-2890| 50| 1| 90| 80| 90| B-|
| George|  Boy|345-67-3901| 40| 1| 11| -1| 4| B|
| Heffalump| Harvey|632-78-9439| 30| 1| 20| 30| 40| C|
=====
```

SparkSQL Scala Command's 1, 2 & 3

Query #1: Max quantity of grades from Test 1 results, order by from least to greatest grade result

CODE:

```
spark.sql("SELECT Test1 AS max FROM df GROUP BY Test1 ORDER BY Test1").show()
```

Query #2: Count of grades from Test 2 column results

CODE:

```
spark.sql("SELECT Test2, count(Grade) AS COUNT FROM df GROUP BY Test2").show()
```

Query #3: Order the grade column in descending order so F grade will appear at the top

CODE:

```
spark.sql("SELECT * FROM df ORDER BY Grade DESC").show()
```

```
santanamonica@dscbigdata: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase

scala> spark.sql("SELECT Test1 AS max FROM df GROUP BY Test1 ORDER BY Test1").show()
+----+
|max|
+----+
| 30|
| 40|
| 41|
| 42|
| 43|
| 44|
| 45|
| 46|
| 47|
| 48|
| 49|
| 50|
+----+

scala> spark.sql("SELECT Test2, count(Grade) AS COUNT FROM df GROUP BY Test2").show()
+-----+
|Test2|COUNT|
+-----+
| 11| 1|
| 78| 1|
| 90| 2|
| 23| 1|
| 97| 1|
| 1| 0|
| 20| 1|
| 80| 1|
+-----+

scala> spark.sql("SELECT * FROM df ORDER BY Grade DESC").show()
+-----+
|Last name|First name|SSN|Test1|Test2|Test3|Test4|Final|Grade|
+-----+
|Noshow|Cecil|345-67-8901|45|11|-1|4|43|F|
|Alfred|Aloysius|123-45-6789|40|90|100|83|49|D-|
|Alfred|University|123-12-1234|41|97|96|97|48|D+|
|Carnivore|Art|1565-89-0123|41|80|60|40|44|C|
|Rubble|Betty|234-56-7890|44|90|80|90|46|C-|
|Dandy|Jim|087-75-4321|47|1|23|36|45|C+|
|Heffalump|Harvey|632-79-9439|30|1|20|30|40|C|
|Gerty|Grama|567-89-0123|41|80|60|40|44|C|
|Elephant|Tia|456-71-9012|45|1|78|88|77|B-|
|Franklin|Benny|234-56-2890|50|1|90|80|90|B-|
|Android|Electric|087-65-4321|42|23|36|45|47|B-|
|Buff|Biff|632-79-9939|46|20|30|40|50|B+|
|George|Boy|345-67-3901|40|1|11|-1|4|B|
|Bumpkin|Fred|456-78-9012|43|78|88|77|45|A-|
|Backus|Jim|143-12-1234|48|1|97|96|97|A+|
|Airpump|Andrew|223-45-6789|49|1|90|100|83|A|
+-----+
```

PySpark Commands

```
santanamonica@dscbigdata: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase

>>> df = spark.read.format('csv').option('header', 'true').load('/data/grades.csv')
>>> df.show()
+-----+
|Last name|First name|SSN|Test1|Test2|Test3|Test4|Final|Grade|
+-----+
|Alfred|Aloysius|123-45-6789|40|90|100|83|49|D-|
|Alfred|University|123-12-1234|41|97|96|97|48|D+|
|Gerty|Grama|567-89-0123|41|80|60|40|44|C|
|Android|Electric|087-65-4321|42|23|36|45|47|B-|
|Bumpkin|Fred|456-78-9012|43|78|88|77|45|A-|
|Rubble|Betty|234-56-7890|44|90|80|90|46|C-|
|Noshow|Cecil|345-67-8901|45|11|-1|4|43|F|
|Buff|Biff|632-79-9939|46|20|30|40|50|B+|
|Airpump|Andrew|223-45-6789|49|1|90|100|83|A|
|Backus|Jim|143-12-1234|48|1|97|96|97|A+|
|Carnivore|Art|1565-89-0123|44|1|80|60|40|D+|
|Dandy|Jim|087-75-4321|47|1|23|36|45|C+|
|Elephant|Tia|456-71-9012|45|1|78|88|77|B-|
|Franklin|Benny|234-56-2890|50|1|90|80|90|B-|
|George|Boy|345-67-3901|40|1|11|-1|4|B|
|Heffalump|Harvey|632-79-9439|30|1|20|30|40|C|
+-----+

>>> df.createOrReplaceTempView('df')
>>> spark.sql("SHOW TABLES").show()
+-----+
|database|tableName|isTemporary|
+-----+
| | df | true |
+-----+

>>> spark.sql("SELECT * FROM df WHERE Final > 50").show()
+-----+
|Last name|First name|SSN|Test1|Test2|Test3|Test4|Final|Grade|
+-----+
|Airpump|Andrew|223-45-6789|49|1|90|100|83|A|
|Backus|Jim|143-12-1234|48|1|97|96|97|A+|
|Elephant|Tia|456-71-9012|45|1|78|88|77|B-|
|Franklin|Benny|234-56-2890|50|1|90|80|90|B-|
+-----+

>>> spark.sql("SELECT * FROM df").show()
+-----+
|Last name|First name|SSN|Test1|Test2|Test3|Test4|Final|Grade|
+-----+
|Alfred|Aloysius|123-45-6789|40|90|100|83|49|D-|
|Alfred|University|123-12-1234|41|97|96|97|48|D+|
|Gerty|Grama|567-89-0123|41|80|60|40|44|C|
|Android|Electric|087-65-4321|42|23|36|45|47|B-|
|Bumpkin|Fred|456-78-9012|43|78|88|77|45|A-|
|Rubble|Betty|234-56-7890|44|90|80|90|46|C-|
|Noshow|Cecil|345-67-8901|45|11|-1|4|43|F|
|Buff|Biff|632-79-9939|46|20|30|40|50|B+|
|Airpump|Andrew|223-45-6789|49|1|90|100|83|A|
|Backus|Jim|143-12-1234|48|1|97|96|97|A+|
|Carnivore|Art|1565-89-0123|44|1|80|60|40|D+|
|Dandy|Jim|087-75-4321|47|1|23|36|45|C+|
|Elephant|Tia|456-71-9012|45|1|78|88|77|B-|
+-----+
```

SparkSQL PySpark Commands 1, 2 & 3

Query #1: Max quantity of grades from Test 1 results, order by from least to greatest grade result

CODE:

```
spark.sql('SELECT Test1 AS max FROM df GROUP BY Test1 ORDER BY Test1').show()
```

Query #2: Count of grades from Test 2 column results

CODE:

```
spark.sql('SELECT Test2, count(Grade) AS COUNT FROM df GROUP BY Test2').show()
```

Query #3: Order the grade column in descending order so F grade will appear at the top

CODE:

```
spark.sql('SELECT * FROM df ORDER BY Grade DESC').show()
```

```
santanamonica@dscbigdata: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
!- Aggregate [Test1#241], [Test1#241 AS max#453]
!- SubqueryAlias df
!- Relation[Last name#238,First name#239,SSN#240,Test1#241,Test2#242,Test3#243,Test4#244,Final#245,Grade#246] csv

>>> spark.sql('SELECT Test1 AS max FROM df GROUP BY Test1 ORDER BY Test1').show()
+----+
|max|
+----+
| 30|
| 40|
| 41|
| 42|
| 43|
| 44|
| 45|
| 46|
| 47|
| 48|
| 49|
| 50|
+----+

>>> spark.sql('SELECT Test2, count(Grade) AS COUNT FROM df GROUP BY Test2').show()
+-----+
|Test2|COUNT|
+-----+
| 11| 1|
| 78| 1|
| 90| 2|
| 23| 1|
| 97| 1|
| 1| 8|
| 20| 1|
| 80| 1|
+-----+

>>> spark.sql('SELECT * FROM df ORDER BY Grade DESC').show()
+-----+
|Last name|First name|SSN|Test1|Test2|Test3|Test4|Final|Grade|
+-----+
|Noshov|Cec|1345-67-8901|45|11|-1|4|43|F| |
|Alfalfa|Aloysius|123-45-6789|40|90|100|83|49|D-|
|Alfred|University|123-12-1234|41|97|96|97|48|D+|
|Carnivore|Art|565-89-0123|44|1|80|60|40|D+|
|Rubble|Betty|234-56-7890|44|90|80|90|46|C-|
|Bandy|Jim|087-75-4321|47|1|23|36|45|C+|
|Hef|Calump|Harvey|632-79-9439|30|1|20|30|40|C|
|Gerty|Gramma|567-89-0123|41|80|60|40|44|C|
|Elephant|Ima|456-71-9012|45|1|78|88|77|B-|
|Franklin|Benny|234-56-7890|50|1|90|80|90|B-|
|Android|Electric|007-65-4321|42|23|36|45|47|D+|
|Butt|Bit|632-79-9439|46|20|30|40|50|D+|
|George|Boy|345-67-8901|40|1|11|-1|4|B|
|Bumpkin|Fred|456-78-9012|43|78|88|77|45|A-|
|Backus|Jim|143-12-1234|48|1|97|96|97|A+|
|Airpump|Andrew|223-45-6789|49|1|90|100|83|A|
+-----+
```

SparkSQL with custom dataset: Sleep dataset, diagnoses of sleep apnea, insomnia or no disorder and features used to diagnose such as demographics and medical information / history

```
santanamonica@dscbigdata: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 6| Normal| 120/80| 70| 8000| 8000| 8000| 8000| 8000| 8000| 8000| 8000|
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
only showing top 20 rows

scala> spark.sql("SELECT * FROM df").show()
-----
|Person ID|Gender|Age| Occupation|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level| BMI Category|Blood Pressure|Heart Rate|Daily Steps|Sleep Disorde
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1| Male| 27| Software Engineer| 6.1| 6| 42| 6| Overweight| 126/83| 77| 4200| Non
| 2| Male| 28| Doctor| 6.2| 6| 60| 8| Normal| 125/80| 75| 10000| Non
| 3| Male| 28| Doctor| 6.2| 6| 60| 8| Normal| 125/80| 75| 10000| Non
| 4| Male| 28| Sales Representative| 5.9| 4| 30| 8| Obese| 140/90| 85| 3000| Sleep Apne
| 5| Male| 28| Sales Representative| 5.9| 4| 30| 8| Obese| 140/90| 85| 3000| Sleep Apne
| 6| Male| 28| Software Engineer| 5.9| 4| 30| 8| Obese| 140/90| 85| 3000| Insomni
| 7| Male| 29| Teacher| 6.3| 6| 40| 7| Obese| 140/90| 82| 3500| Insomni
| 8| Male| 29| Doctor| 7.8| 7| 75| 6| Normal| 120/80| 70| 8000| Non
| 9| Male| 29| Doctor| 7.8| 7| 75| 6| Normal| 120/80| 70| 8000| Non
| 10| Male| 29| Doctor| 7.8| 7| 75| 6| Normal| 120/80| 70| 8000| Non
| 11| Male| 29| Doctor| 6.1| 6| 30| 8| Normal| 120/80| 70| 8000| Non
| 12| Male| 29| Doctor| 7.8| 7| 75| 6| Normal| 120/80| 70| 8000| Non
| 13| Male| 29| Doctor| 6.1| 6| 30| 8| Normal| 120/80| 70| 8000| Non
| 14| Male| 29| Doctor| 6| 6| 30| 8| Normal| 120/80| 70| 8000| Non
| 15| Male| 29| Doctor| 6| 6| 30| 8| Normal| 120/80| 70| 8000| Non
| 16| Male| 29| Doctor| 6| 6| 30| 8| Normal| 120/80| 70| 8000| Non
| 17| Female| 29| Nurse| 6.5| 5| 40| 7| Normal Weight| 132/87| 80| 4000| Sleep Apne
| 18| Male| 29| Doctor| 6| 6| 30| 8| Normal| 120/80| 70| 8000| Sleep Apne
| 19| Female| 29| Nurse| 6.5| 5| 40| 7| Normal Weight| 132/87| 80| 4000| Insomni
| 20| Male| 30| Doctor| 7.6| 7| 75| 6| Normal| 120/80| 70| 8000| Non
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
only showing top 20 rows

scala>
```

SparkSQL Scala Command's 1, 2 & 3 with custom dataset

Query #1: Max quantity of occupation titles from occupation column results, and order by to make it alphabetical order

CODE:

```
spark.sql("SELECT Occupation AS max FROM df GROUP BY Occupation ORDER BY Occupation").show()
```

Query #2: Count of how many people in the dataset per occupation from occupation column results

CODE:

```
spark.sql("SELECT Occupation, count(Gender) AS COUNT FROM df GROUP BY Occupation").show()
```

CODE:

```

santanamonica@dscbigneta: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
scala> spark.sql("SELECT Occupation AS max FROM df GROUP BY Occupation ORDER BY Occupation").show()
-----+
| max|
-----+
| Accountant|
| Doctor|
| Engineer|
| Lawyer|
| Manager|
| Nurse|
| Sales Representative|
| Salesperson|
| Scientist|
| Software Engineer|
| Teacher|
-----+

scala> spark.sql("SELECT Occupation, count(Gender) AS COUNT FROM df GROUP BY Occupation").show()
-----+
| Occupation|COUNT|
-----+
| Scientist| 4|
| Nurse| 73|
| Salesperson| 32|
| Lawyer| 47|
| Teacher| 40|
| Sales Representative| 21|
| Doctor| 71|
| Engineer| 63|
| Accountant| 37|
| Manager| 11|
| Software Engineer| 41|
-----+

scala> spark.sql("SELECT * FROM df ORDER BY Age DESC").show()
-----+
| Person ID|Gender|Age|Occupation|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|BMI Category|Blood Pressure|Heart Rate|Daily Steps|Sleep Disorder|
-----+
| 361|Female|59|Nurse| 8.2| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 364|Female|59|Nurse| 8.2| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 364|Female|59|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 362|Female|59|Nurse| 8.2| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 359|Female|59|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| None|
| 369|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 370|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 371|Female|59|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 372|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 373|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 374|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 366|Female|59|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 367|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 360|Female|59|Nurse| 8.1| 9| 75| 3| Overweight| 140/95| 68| 7000| None|
| 368|Female|59|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 363|Female|59|Nurse| 8.2| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 355|Female|58|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
| 353|Female|58|Nurse| 8| 9| 75| 3| Overweight| 140/95| 68| 7000| Sleep Apnea|
-----+

```

The main difference I see using SparkSQL compared to DataFrames is the efficiency. The dataset I chose is not a small one, and creating a dataframe with it took longer than uploading it in SparkSQL. Even the commands outputs where the entire dataset would be reordered, appeared in rapid speed. Granted it was just the first 20 rows, but I know it still would appear faster than running the dataframe transformations in jupyter notebook for example with the same `head(20)` rows.