QUERYING / ANALYZING DATAFRAME

SELECT AND WHERE TRANSFORMATIONS

On Scala

FIRST STEP TO DO

Create 3 DataFrames by name **empDF, deptDF**, and **salgradeDF** using **EMP, DEPT, and SALGRADE** files which you have saved in JSON format in "TEST YOURSELF" exercise.

HINT: **hdfs dfs -ls /user/root/payroll**

1. Count the number of rows in EMP.

   empDF.count()

2. DataFrame transformations typically return another DataFrame. Try using a select transformation to return a DataFrame with only the empno and salary columns from empDF, then display its schema. Note that only the selected columns are in the schema.

scala> **val salaryDF = empDF.select("empno","salary")**

scala> **salaryDF.printSchema**

scala> **salaryDF.show**

3. Using the where clause to filter

scala> **val highsalaryDF = salaryDF.where("salary > 2000")**

scala> **highsalaryDF.show()**

4. Transformations in a query can be chained together. Execute a single command to show the results of a query using select and where. The resulting DataFrame will contain the employees having salary > 2000.

scala> **val highsalaryDF=spark.read.json("/user/root/payroll/empjson/").select("empno","salary").where("salary > 2000")**

```
scala> highsalaryDF.show
+-----+------+
|empno|salary|
+-----+------+
| 7566|  2975|
| 7698|  2850|
| 7782|  2450|
| 7788|  3000|
| 7839|  5000|
| 7902|  3000|
+-----+------+
```

On Python

FIRST STEP TO DO

Create 3 DataFrames by name **empDF, deptDF, and salgradeDF** using **EMP, DEPT, and SALGRADE** files which you have saved in JSON format in "TEST YOURSELF" exercise.

HINT: **hdfs dfs -ls /user/root/payroll**

1. Count the number of rows in EMP.

    empDF.count()

2. DataFrame transformations typically return another DataFrame. Try using a select transformation to return a DataFrame with only the empno and salary columns from empDF, then display its schema. Note that only the selected columns are in the schema.

>>> **salaryDF = empDF.select("empno","salary")**

>>> **salaryDF.printSchema**

>>> **salaryDF.show**

3. Using the where clause to filter

>>> **highsalaryDF = salaryDF.where("salary > 2000")**

>>> **highsalaryDF.show()**

4. Transformations in a query can be chained together. Execute a single command to show the results of a query using select and where. The resulting DataFrame will contain the employees having salary > 2000.

>>>
highsalaryDF=spark.read.json("/user/root/payroll/empjson/").select("empno","salary").where("salary > 2000")

```
>>> highsalaryDF.show()
+-----+------+
|empno|salary|
+-----+------+
| 7566|  2975|
| 7698|  2850|
| 7782|  2450|
| 7788|  3000|
| 7839|  5000|
| 7902|  3000|
+-----+------+
```

COLUMNS, COLUMN NAMES, AND COLUMN EXPRESSIONS

REFERING A COLUMN

On Scala

1. COLUMN NAME WITH DATAFRAME
   scala> **empDF.select(empDF("ENAME")).show**

2. REFERING A COLUMN WITH $
   scala> **empDF.select($"ENAME").show**

3. REFERING A COLUMN WITH '
   scala> **empDF.select('ENAME).show**

On Python

1. COLUMN NAME WITH DATAFRAME
   >>> **empDF.select(empDF.ENAME).show()**

2. REFERING A COLUMN WITH []
   >>> **empDF.select(empDF['ENAME']).show()**

COLUMN EXPRESSIONS

ARTHEMATIC OPERATORS: + , - , %, /, and *

COMPARATIVE AND LOGICAL: <, >, &&, and ||

EQUALITY OPERATOR: SCALA: === ;**PYTHON:** ==

STRING FUNCTIONS: contains, like and substring

DATA TESTING: isnull, isNotNull, and NaN (not a number)

SORTING: orderBy: asc and desc

On Scala

1. Increment the salary of Employees by 5%
   scala> **val NewSalary=empDF.select($"ENAME",$"SALARY", $"SALARY" * .05).show**

```
+------+------+--------------+
| ENAME|SALARY|(SALARY * 0.05)|
+------+------+--------------+
| SMITH|   800|          40.0|
| ALLEN|  1600|          80.0|
|  WARD|  1250|          62.5|
| JONES|  2975|        148.75|
|MARTIN|  1250|          62.5|
| BLAKE|  2850|         142.5|
| CLARK|  2450|         122.5|
| SCOTT|  3000|         150.0|
|  KING|  5000|         250.0|
|TURNER|  1500|          75.0|
| ADAMS|  1100|          55.0|
| JAMES|   950|          47.5|
|  FORD|  3000|         150.0|
|MILLER|  2000|         100.0|
+------+------+--------------+
```

2. Show employees whose names starts with "A"
   scala> **empDF.where(empDF("ENAME").startsWith("A")).show**

```
+----+------+-----+-----+---------+---------+-------+------+
|COMM|DEPTNO|EMPNO|ENAME| HIREDATE|      JOB|MGRCODE|SALARY|
+----+------+-----+-----+---------+---------+-------+------+
| 300|    30| 7499|ALLEN|20-Feb-81|SALESMAN|   7698|  1600|
|   0|    20| 7876|ADAMS|23-May-87|   CLERK|   7788|  1100|
+----+------+-----+-----+---------+---------+-------+------+
```


3. Using column Alias
   scala> **val salary5percent=empDF.select($"ENAME",$"SALARY",($"SALARY" * .05).alias("NEW_SALARY"))**

   scala> **val newsalary=salary5percent.select($"ENAME",($"SALARY" + $"NEW_SALARY").alias("INCREASED_SAL"))**

```
scala> newsalary.show
+------+-------------+
| ENAME|INCREASED_SAL|
+------+-------------+
| SMITH|        840.0|
| ALLEN|       1680.0|
|  WARD|       1312.5|
| JONES|      3123.75|
|MARTIN|       1312.5|
| BLAKE|       2992.5|
| CLARK|       2572.5|
| SCOTT|       3150.0|
|  KING|       5250.0|
|TURNER|       1575.0|
| ADAMS|       1155.0|
| JAMES|        997.5|
|  FORD|       3150.0|
|MILLER|       2100.0|
+------+-------------+
```

4. Sorting the data (By Salary ascending and descending)
   scala> **import org.apache.spark.sql.functions._**

   scala> **empDF.orderBy(asc("SALARY")).show**

   scala> **empDF.orderBy(desc("SALARY")).show**

On Python

1. Increment the salary of Employees by 5%
   >>> **empDF.select("ENAME",empDF.SALARY * .05).show()**

2. Show employees whose names starts with "A"

   >>> **empDF.where(empDF.ENAME.startswith("A")).show()**

3. Using column Alias

   >>> **salary5percent=empDF.select("ENAME","SALARY",(empDF.SALARY * .05).alias("NEW_SALARY"))**

   >>> **newsalary=salary5percent.select("ENAME",(salary5percent.SALARY + salary5percent.NEW_SALARY).alias("INCREASED_SAL"))**

```
>>> newsalary.show()
+------+-------------+
| ENAME|INCREASED_SAL|
+------+-------------+
| SMITH|        840.0|
| ALLEN|       1680.0|
|  WARD|       1312.5|
| JONES|      3123.75|
|MARTIN|       1312.5|
| BLAKE|       2992.5|
| CLARK|       2572.5|
| SCOTT|       3150.0|
|  KING|       5250.0|
|TURNER|       1575.0|
| ADAMS|       1155.0|
| JAMES|        997.5|
|  FORD|       3150.0|
|MILLER|       2100.0|
+------+-------------+
```

4. Sorting the data (By Salary ascending and descending)

   >>> **from pyspark.sql.functions import ***

   >>> **empDF.orderBy(asc("SALARY")).show()**

   >>> **empDF.orderBy(desc("SALARY")).show()**

GROUPING THE DATA

groupBy clause and Group Functions

COUNT

MIN

MAX

MEAN (AVG)

SUM

PIVOT

AGG

Ex:

On Scala

scala> **empDF.groupBy($"deptno").count().show**

On Python

>>> **empDF.groupBy("DEPTNO").count().show()**

MAX and MIN

On Scala

scala> **empDF.groupBy().max("SALARY").show**

```
+-----------+
|max(SALARY)|
+-----------+
|       5000|
+-----------+
```

scala> **empDF.groupBy().min("SALARY").show**

```
+-----------+
|min(SALARY)|
+-----------+
|        800|
+-----------+
```

On Python

>>> **empDF.groupBy().max("SALARY").show()**

```
+-----------+
|max(SALARY)|
+-----------+
|       5000|
+-----------+
```



```
>>> empDF.groupBy().min("SALARY").show()
```

```
+-----------+
|min(SALARY)|
+-----------+
|        800|
+-----------+
```

DEPARTMENT WISE SUM OF SALARIES

On Python

```
>>> empDF.groupBy("DEPTNO").sum("SALARY").show()
```

```
+------+-----------+
|DEPTNO|sum(SALARY)|
+------+-----------+
|    10|       9450|
|    30|       9400|
|    20|      10875|
+------+-----------+
```

On Scala

```
scala> empDF.groupBy("DEPTNO").sum("SALARY").show
```

```
+------+-----------+
|DEPTNO|sum(SALARY)|
+------+-----------+
|    10|       9450|
|    30|       9400|
|    20|      10875|
+------+-----------+
```


JOINING THE DATAFRAMES

INNER

OUTER

LEFT_OUTER

RIGHT_OUTER

LEFTSEMI

CROSS JOIN

Example

We have 2 dataframes empDF and deptDF. In empDF we have employees information along with their department numbers, and in deptDF we have department names and locations of the corresponding department numbers.

Now we need report showing every employee's details along with their department names and locations. Hence we need to join the both dataframes.

The common column among the both dataframes is "deptno"

On Scala

scala> **val empdeptjoinDF=empDF.join(deptDF,empDF("DEPTNO") === deptDF("DEPTNO")).show**

```
+----+------+-----+------+---------+---------+-------+------+------+----------+--------+
|COMM|DEPTNO|EMPNO| ENAME| HIREDATE|      JOB|MGRCODE|SALARY|DEPTNO|     DNAME|LOCATION|
+----+------+-----+------+---------+---------+-------+------+------+----------+--------+
|   0|    20| 7369| SMITH|17-Dec-80|    CLERK|   7902|   800|    20|  RESEARCH|  DALLAS|
| 300|    30| 7499| ALLEN|20-Feb-81| SALESMAN|   7698|  1600|    30|     SALES| CHICAGO|
| 500|    30| 7521|  WARD|22-Feb-81| SALESMAN|   7698|  1250|    30|     SALES| CHICAGO|
|   0|    20| 7566| JONES|02-Apr-81|  MANAGER|   7839|  2975|    20|  RESEARCH|  DALLAS|
|1400|    30| 7654|MARTIN|28-Sep-81| SALESMAN|   7698|  1250|    30|     SALES| CHICAGO|
|   0|    30| 7698| BLAKE|01-May-81|  MANAGER|   7839|  2850|    30|     SALES| CHICAGO|
|   0|    10| 7782| CLARK|09-Jun-81|  MANAGER|   7839|  2450|    10|ACCOUNTING|NEW YORK|
|   0|    20| 7788| SCOTT|19-Apr-87|  ANALYST|   7566|  3000|    20|  RESEARCH|  DALLAS|
|   0|    10| 7839|  KING|17-Nov-81|PRESIDENT|      0|  5000|    10|ACCOUNTING|NEW YORK|
|   0|    30| 7844|TURNER|08-Sep-81| SALESMAN|   7698|  1500|    30|     SALES| CHICAGO|
|   0|    20| 7876| ADAMS|23-May-87|    CLERK|   7788|  1100|    20|  RESEARCH|  DALLAS|
|   0|    30| 7900| JAMES|03-Dec-81|    CLERK|   7698|   950|    30|     SALES| CHICAGO|
|   0|    20| 7902|  FORD|03-Dec-81|  ANALYST|   7566|  3000|    20|  RESEARCH|  DALLAS|
|   0|    10| 7934|MILLER|23-Jan-82|    CLERK|   7782|  2000|    10|ACCOUNTING|NEW YORK|
+----+------+-----+------+---------+---------+-------+------+------+----------+--------+
```


On Python

>>> empdeptjoinDF=empDF.join(deptDF, "DEPTNO")

```
>>> empdeptjoinDF.show()
+------+----+-----+------+---------+---------+-------+------+----------+--------+
|DEPTNO|COMM|EMPNO| ENAME| HIREDATE|      JOB|MGRCODE|SALARY|     DNAME|LOCATION|
+------+----+-----+------+---------+---------+-------+------+----------+--------+
|    20|   0| 7369| SMITH|17-Dec-80|    CLERK|   7902|   800|  RESEARCH|  DALLAS|
|    30| 300| 7499| ALLEN|20-Feb-81| SALESMAN|   7698|  1600|     SALES| CHICAGO|
|    30| 500| 7521|  WARD|22-Feb-81| SALESMAN|   7698|  1250|     SALES| CHICAGO|
|    20|   0| 7566| JONES|02-Apr-81|  MANAGER|   7839|  2975|  RESEARCH|  DALLAS|
|    30|1400| 7654|MARTIN|28-Sep-81| SALESMAN|   7698|  1250|     SALES| CHICAGO|
|    30|   0| 7698| BLAKE|01-May-81|  MANAGER|   7839|  2850|     SALES| CHICAGO|
|    10|   0| 7782| CLARK|09-Jun-81|  MANAGER|   7839|  2450|ACCOUNTING|NEW YORK|
|    20|   0| 7788| SCOTT|19-Apr-87|  ANALYST|   7566|  3000|  RESEARCH|  DALLAS|
|    10|   0| 7839|  KING|17-Nov-81|PRESIDENT|      0|  5000|ACCOUNTING|NEW YORK|
|    30|   0| 7844|TURNER|08-Sep-81| SALESMAN|   7698|  1500|     SALES| CHICAGO|
|    20|   0| 7876| ADAMS|23-May-87|    CLERK|   7788|  1100|  RESEARCH|  DALLAS|
|    30|   0| 7900| JAMES|03-Dec-81|    CLERK|   7698|   950|     SALES| CHICAGO|
|    20|   0| 7902|  FORD|03-Dec-81|  ANALYST|   7566|  3000|  RESEARCH|  DALLAS|
|    10|   0| 7934|MILLER|23-Jan-82|    CLERK|   7782|  2000|ACCOUNTING|NEW YORK|
+------+----+-----+------+---------+---------+-------+------+----------+--------+
```

OUTER JOIN

On Python

>>> empdeptjoinDF=empDF.join(deptDF,"DEPTNO",
"left_outer").show()

On Scala

Scala> val
empdeptjoinDF=empDF.join(deptDF,empDF("DEPTNO") ===
deptDF("DEPTNO"),"left_outer"
).show

JOIN USING DIFFERENT COLUMN NAMES

On Scala: empDF.join(deptDF, $"deptno" === $"dno").show

On Python: empDF.join(deptDF,empDF.deptno == deptDF.dno)

