**•Install python pip**

  sudo apt-get install python-pip

**•Install pyspark using pip**

  sudo python –m pip --no-cache-dir install pyspark

**•Install JRE**

  sudo apt install default-jre

**•Export JAVA\_HOME in .bashrc file**

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-i386

**•Apply .bashrc file**

  source ~/.bashrc

**#How to apply**

source ~/.bashrc

<https://github.com/shyam-kantesariya/big_data_course>

>>> tx1 = txF.map(lambda x: Row(account\_id=x.split(",")[0], amt=int(x.split(",")[1])))

>>> tx1

PythonRDD[27] at RDD at PythonRDD.scala:53

>>> tx1.take(1)

[Row(account\_id=u'0', amt=3525)]

>>> tx1 = txF.map(lambda x: Row(account\_id=x.split(",")[0], amt=int(x.split(",")[1]))).toDF()

>>> tx1.show()

+----------+-----+

|account\_id|  amt|

+----------+-----+

|         0| 3525|

|         1|-6955|

|         2|-7983|

|         3| 5839|

|         4|-7184|

|         5| 2000|

|         6| 5012|

|         7| 6177|

|         8|-5843|

|         9|-6765|

|        10|-5252|

|        11| 6685|

|        12| 5679|

|        13|-6845|

|        14| 1987|

|        15|-1752|

|        16| -638|

|        17| 6893|

|        18|-6002|

|        19| -121|

+----------+-----+

only showing top 20 rows

>>> tx1.printSchema()

root

 |-- account\_id: string (nullable = true)

 |-- amt: long (nullable = true)

>>> bal1 = balF.map(lambda x: Row(account\_id=x.split(",")[0], balance = int(x.split(",")[1]))).toDF()

>>> bal1.printSchema()

root

 |-- account\_id: string (nullable = true)

 |-- balance: long (nullable = true)

>>> tx2 = tx1.groupBy("account\_id").agg(sum("amt")).alias("bal")

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: unsupported operand type(s) for +: 'int' and 'str'

>>> tx2 = tx1.groupBy("account\_id").agg(sum("amt").alias("bal"))

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: unsupported operand type(s) for +: 'int' and 'str'

>>> sum(1,2)

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: 'int' object is not iterable

>>> sum(1)

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: 'int' object is not iterable

>>> help(sum)

>>> sum((1,2))

3

>>> sum((1,5))

6

>>> from pyspark.sql.functions import sum as \_sum

>>> tx2 = tx1.groupBy("account\_id").agg(\_sum("amt").alias("bal"))

>>> tx2.show()

+----------+------+

|account\_id|   bal|

+----------+------+

|         7| 27262|

|        51|   -11|

|        15| 10754|

|        54| 13096|

|        11|  6243|

|        29|-11870|

|        69|  5798|

|        42| 10111|

|        73| -4855|

|        87| -1853|

|        64| 11922|

|         3| -3023|

|        30| -3866|

|        34|  3276|

|        59| -6333|

|         8| 11665|

|        22| -4481|

|        28| -9851|

|        85|  5367|

|        16| -3976|

+----------+------+

only showing top 20 rows

>>> tx2.printSchema

<bound method DataFrame.printSchema of DataFrame[account\_id: string, bal: bigint]>

>>> tx2.printSchema()

root

 |-- account\_id: string (nullable = true)

 |-- bal: long (nullable = true)

>>> bal1.printSchema()

root

 |-- account\_id: string (nullable = true)

 |-- balance: long (nullable = true)

>>> joinedDf = tx2.join(bal1, tx2.account\_id=bal1.account\_id)

  File "<stdin>", line 1

SyntaxError: keyword can't be an expression

>>> joinedDf = tx2.join(bal1, tx2.account\_id==bal1.account\_id)

>>> joinedDf.show(1)

+----------+---+----------+-------+

|account\_id|bal|account\_id|balance|

+----------+---+----------+-------+

|        51|-11|        51|    -11|

+----------+---+----------+-------+

only showing top 1 row

>>> joinedDf = tx2.join(bal1, account\_id)

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

NameError: name 'account\_id' is not defined

>>> joinedDf = tx2.join(bal1, "account\_id")

>>> joinedDf.show(1)

+----------+---+-------+

|account\_id|bal|balance|

+----------+---+-------+

|        51|-11|    -11|

+----------+---+-------+

only showing top 1 row

>>> errorAccounts = joinedDf.filter(joinedDf.bal != joinedDf.balance)

>>> errorAccounts.show(5)

+----------+------+-------+

|account\_id|   bal|balance|

+----------+------+-------+

|         7| 27262|   2726|

|        31|-10770|  -1077|

|        78|-10615|  -1615|

|        40| 15900|   1500|

|        44|-10552|  -1552|

+----------+------+-------+

only showing top 5 rows

>>> errorAccounts.rdd.saveAsTextFile("result1")

>>> errorAccounts.rdd.map(lambda x: str(x[0]) + "," + str(x[1]) + "," + str(x[2]).saveAsTextFile("result2")

... )

PythonRDD[109] at RDD at PythonRDD.scala:53

>>> errorAccounts.rdd.map(lambda x: str(x[0]) + "," + str(x[1]) + "," + str(x[2])).saveAsTextFile("result2")

>>> errorAccounts.rdd.map(lambda x: str(x[0]) + "," + str(x[1]) + "," + str(x[2])).repartition(1).saveAsTextFile("result3")

>>> errorAccounts.save("dfresult1")

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/dataframe.py", line 1301, in \_\_getattr\_\_

    "'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'save'

>>> errorAccounts.save("dfresult1")

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/dataframe.py", line 1301, in \_\_getattr\_\_

    "'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'save'

>>> errorAccounts.write.save("dfresult1")

>>> errorAccounts.write.format("json").save("dfresult2")

Databricks community cluster:

<https://community.cloud.databricks.com/login.html>

**Install mysql on ubuntu**

sudo apt-get update

sudo apt-get install mysql-server

## Set ubuntu password

sudo passwd ubuntu

## Start mysql server

systemctl start mysql

#Enable mysql remote access outside EC2

Step 1: Modify /etc/mysql/my.cnf file as following. Add following two lines. there is a tab between key and = sign

[mysqld]

bind-address    = 0.0.0.0

Step 2: Restart mysql server

sudo /etc/init.d/mysql restart

systemctl status mysql.service

Step 3: Create a new user as following on mysql prompt

CREATE USER 'non-root-user'@'localhost' IDENTIFIED BY 'any\_password\_u\_like';

CREATE USER 'non-root-user'@'%' IDENTIFIED BY 'any\_password\_u\_like';

GRANT ALL ON \*.\* TO 'non-root-user'@'localhost';

GRANT ALL ON \*.\* TO 'non-root-user'@'%';

**#### READ mysql data into a dataframe**

jdbcDF = spark.read \

    .format("jdbc") \

    .option("url", "jdbc:mysql://35.183.177.131:3306") \

    .option("dbtable", "bigdata.t1") \

    .option("user", "spark") \

    .option("password", "spark") \

    .load()

jdbcDF.show()

jdbcDF.write \

    .jdbc("jdbc:mysql://35.183.177.131:3306", "bigdata.t2", mode="append", properties={"user": "spark", "password": "spark"})

**Using Spark SQL:**

>>> tx1.registerTempTable("tranx")

>>> bal1.registerTempTable("accBal")

>>> tx2 = sqlContext.sql("select account\_id, cast(sum(amt) as int) as bal from tranx group by account\_id")

20/01/18 10:34:48 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0

20/01/18 10:34:48 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException

20/01/18 10:34:48 WARN ObjectStore: Failed to get database global\_temp, returning NoSuchObjectException

>>>

>>> tx2.registerTempTable("aggTranxBal")

>>> joinedDf = sqlContext.sql("select b.account\_id, t.bal, b.balance from aggTranxbal t join accBal b on t.account\_id = b.account\_id" )

>>> joinedDf.registerTempTable("joinedDf")

>>> errorAccounts = sqlContext.sql("select \* from joinedDf j where j.bal != j.balance" )

>>> errorAccounts.show()

+----------+------+-------+

|account\_id|   bal|balance|

+----------+------+-------+

|         7| 27262|   2726|

|        31|-10770|  -1077|

|        78|-10615|  -1615|

|        40| 15900|   1500|

|        44|-10552|  -1552|

|        86| -4834|   -483|

|        10|-18621|  -1621|

|        66|  6475|    647|

|        91|  2813|    283|

|        94| -5249|   -524|

|        72| -6658|   -658|

+----------+------+-------+

**Multiple ways of creating dataframe**

>>> txF

/home/teacher/Documents/big\_data\_course-master/Spark/lecture1/data/transactions.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0

>>> txF.take(1)

[u'0,3525']

>>> tx1 = txF.map(lambda x: x.split(","))

>>> tx1.take(1)

[[u'0', u'3525']]

>>> sqlContext

<pyspark.sql.context.SQLContext object at 0xb7375e0c>

>>> df1 = sqlContext.createDataFrame(tx1)

>>> df.show()

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df1.show(3)

+---+-----+

| \_1|   \_2|

+---+-----+

|  0| 3525|

|  1|-6955|

|  2|-7983|

+---+-----+

only showing top 3 rows

>>> df1.printSchema()

root

 |-- \_1: string (nullable = true)

 |-- \_2: string (nullable = true)

>>> df1 = sqlContext.createDataFrame(tx1, ['account\_id','balance'])

>>> df1.show(3)

+----------+-------+

|account\_id|balance|

+----------+-------+

|         0|   3525|

|         1|  -6955|

|         2|  -7983|

+----------+-------+

only showing top 3 rows

>>> df1.printSchema()

root

 |-- account\_id: string (nullable = true)

 |-- balance: string (nullable = true)

>>> from pyspark.sql import Row

>>> account = Row('account\_id', 'balance')

>>> tx2 = tx1.map(lambda x: account(\*x))

>>> df1 = sqlContext.createDataFrame(tx2)

>>> df1.printSchema()

root

 |-- account\_id: string (nullable = true)

 |-- balance: string (nullable = true)

>>> df1.show(3)

+----------+-------+

|account\_id|balance|

+----------+-------+

|         0|   3525|

|         1|  -6955|

|         2|  -7983|

+----------+-------+

only showing top 3 rows

>>> from pyspark.sql.types import \*

>>> tx2 = tx1.map(lambda x: (int(x[0]),int(x[1])))

>>> schema = StructType([StructField("account\_id", IntegerType(), True), StructField("balance", IntegerType(), True)])

>>> df1 = sqlContext.createDataFrame(tx2,schema)

>>> df1.show(3)+----------+-------+

|account\_id|balance|

+----------+-------+

|         0|   3525|

|         1|  -6955|

|         2|  -7983|

+----------+-------+

only showing top 3 rows

>>> df1.printSchema()root

 |-- account\_id: integer (nullable = true)

 |-- balance: integer (nullable = true)

>>> tx1 = txF.map(lambda x: x.split(","))

>>> tx2 = tx1.map(lambda x: (int(x[0]),int(x[1])))

>>> df1 = sqlContext.createDataFrame(tx2, ['account\_id', 'balance'])

>>> df1.printSchema()

root

 |-- account\_id: long (nullable = true)

 |-- balance: long (nullable = true)

**Different ways of reading writing files:**

>>> myjson = spark.read.json("/home/teacher/Documents/big\_data\_course-master/Spark/lecture1/data/emp.json")

>>> myjson

DataFrame[deptId: bigint, firstName: string, lastName: string]

>>> myjson.show()

+------+---------+--------+

|deptId|firstName|lastName|

+------+---------+--------+

|     1|     John|     Doe|

+------+---------+--------+

>>> myjson.show(3)

+------+---------+--------+

|deptId|firstName|lastName|

+------+---------+--------+

|     1|     John|     Doe|

+------+---------+--------+

>>> myjson.show()

+------+---------+--------+

|deptId|firstName|lastName|

+------+---------+--------+

|     1|     John|     Doe|

+------+---------+--------+

>>> myjson = spark.read.json("/home/teacher/Documents/big\_data\_course-master/Spark/lecture1/data/emp.json")

>>> myjson.show()

+------+---------+--------+

|deptId|firstName|lastName|

+------+---------+--------+

|     1|     John|     Doe|

|     1|     Anna|   Smith|

|     2|    Peter|   Jones|

+------+---------+--------+

>>> x = spark.read.load("/home/teacher/Documents/big\_data\_course-master/Spark/lecture1/data/emp.csv", format='com.databricks.spark.csv',header='true',inferSchema='true')

>>> x.show()

+---------+--------+------+

|firstName|lastName|deptId|

+---------+--------+------+

|     John|     Doe|     1|

|     Anna|   Smith|     1|

|    Peter|   Jones|     2|

+---------+--------+------+

>>> x

DataFrame[firstName: string, lastName: string, deptId: int]

>>> myjson.write.format("json").save("jsonresult")

>>> x.write.csv("csvresult")

>>> dat = spark.read.load("/home/teacher/Documents/big\_data\_course-master/Spark/lecture1/data/emp.dat", format="csv", sep='|', inferSchema="true",header="true")

>>> dat.show()

+---------+--------+------+

|firstName|lastName|deptId|

+---------+--------+------+

|     John|     Doe|     1|

|     Anna|   Smith|     1|

|    Peter|   Jones|     2|

+---------+--------+------+

>>> dat.write.csv("datresult","|")

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/readwriter.py", line 923, in csv

    self.mode(mode)

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/readwriter.py", line 596, in mode

    self.\_jwrite = self.\_jwrite.mode(saveMode)

  File "/usr/local/lib/python2.7/dist-packages/pyspark/python/lib/py4j-0.10.7-src.zip/py4j/java\_gateway.py", line 1257, in \_\_call\_\_

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/utils.py", line 79, in deco

    raise IllegalArgumentException(s.split(': ', 1)[1], stackTrace)

pyspark.sql.utils.IllegalArgumentException: u"Unknown save mode: |. Accepted save modes are 'overwrite', 'append', 'ignore', 'error', 'errorifexists'."

>>> dat.write.csv("datresult",'|')

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/readwriter.py", line 923, in csv

    self.mode(mode)

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/readwriter.py", line 596, in mode

    self.\_jwrite = self.\_jwrite.mode(saveMode)

  File "/usr/local/lib/python2.7/dist-packages/pyspark/python/lib/py4j-0.10.7-src.zip/py4j/java\_gateway.py", line 1257, in \_\_call\_\_

  File "/usr/local/lib/python2.7/dist-packages/pyspark/sql/utils.py", line 79, in deco

    raise IllegalArgumentException(s.split(': ', 1)[1], stackTrace)

pyspark.sql.utils.IllegalArgumentException: u"Unknown save mode: |. Accepted save modes are 'overwrite', 'append', 'ignore', 'error', 'errorifexists'."

>>> dat.write.csv("datresult",'|', mode='overwrite')

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: csv() got multiple values for keyword argument 'mode'

>>> help(dat.write.csv)

>>> dat.write.csv("datresult",sep='|', mode='overwrite')

>>> x = spark.read.parquet("dfresult2/\*")

>>> x.show()

+----------+------+-------+

|account\_id|   bal|balance|

+----------+------+-------+

|        91|  2813|    283|

|        94| -5249|   -524|

|        66|  6475|    647|

|        10|-18621|  -1621|

|        78|-10615|  -1615|

|        86| -4834|   -483|

|        40| 15900|   1500|

|        31|-10770|  -1077|

|        44|-10552|  -1552|

|        72| -6658|   -658|

|         7| 27262|   2726|

+----------+------+-------+

>>> help(dat.write.csv)

>>> df = spark.sql("SELECT \* FROM parquet.`dfresult2/\*`")

20/01/18 11:06:20 WARN ObjectStore: Failed to get database parquet, returning NoSuchObjectException

>>> df = spark.sql("SELECT \* FROM parquet.`dfresult2/\*/`")

20/01/18 11:06:42 WARN ObjectStore: Failed to get database parquet, returning NoSuchObjectException

>>> df.show()

+----------+------+-------+

|account\_id|   bal|balance|

+----------+------+-------+

|        91|  2813|    283|

|        94| -5249|   -524|

|        66|  6475|    647|

|        10|-18621|  -1621|

|        78|-10615|  -1615|

|        86| -4834|   -483|

|        40| 15900|   1500|

|        31|-10770|  -1077|

|        44|-10552|  -1552|

|        72| -6658|   -658|

|         7| 27262|   2726|

+----------+------+-------+

>>> df = spark.sql("SELECT \* FROM parquet.`/home/teacher/dfresult2/\*` where balance=283")

20/01/18 11:07:51 WARN ObjectStore: Failed to get database parquet, returning NoSuchObjectException

>>> df.show()

+----------+----+-------+

|account\_id| bal|balance|

+----------+----+-------+

|        91|2813|    283|

+----------+----+-------+