pySpark Cheatsheet

Table of Contents

[Command Shell Execution Within iPython 1](#_Toc504612895)

[Examine Underlying HDFS Directory Structures in a Jupyter Notebook 1](#_Toc504612896)

[Transfer a CSV file to HDFS 3](#_Toc504612897)

[DataFrames (pyspark.sql.dataframe.DataFrame) 4](#_Toc504612898)

[Concatenate 2 pySpark DataFrames, row-wise 4](#_Toc504612899)

[Display DataFrame Columns and Types 4](#_Toc504612900)

[DataFrame Schema 5](#_Toc504612901)

[Extract a column from a pyspark DataFrame as a List 5](#_Toc504612902)

[Sample a DataFrame (approximate sample) 5](#_Toc504612903)

[Sort a DataFrame 5](#_Toc504612904)

[pyspark.sql 6](#_Toc504612905)

# Command Shell Execution Within iPython



## Examine Underlying HDFS Directory Structures in a Jupyter Notebook

%sx hdfs dfs -ls ../../lambda

Output:

['Found 6 items',

'drwxr-xr-x - saviops saviops 0 2017-08-09 16:59 ../../lambda/UNKNOWN',

'drwxr-xr-x - saviops saviops 0 2017-08-23 03:44 ../../lambda/definitions-parquet',

'drwxr-xr-x - saviops saviops 0 2017-07-31 13:06 ../../lambda/smf-json',

'drwxr-xr-x - saviops saviops 0 2017-08-23 03:25 ../../lambda/smf-parquet',

'drwxr-xr-x - hdfs saviops 0 2017-06-28 15:44 ../../lambda/summaries-json',

'drwxr-xr-x - saviops saviops 0 2017-08-23 04:11 ../../lambda/summaries-parquet']

Logout[Control Panel](https://10.200.4.13/hub/home)Untitled Last Checkpoint: a few seconds ago (autosaved)

pySpark Basic (Spark 1.4.1)

* [File](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)
* [Edit](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)
* [View](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)
* [Insert](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)
* [Cell](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)
* [Kernel](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)
* [Help](https://10.200.4.13/user/bbeauchamp/notebooks/Untitled.ipynb)

CodeMarkdownRaw NBConvertHeading

Cell Toolbar:NoneEdit MetadataRaw Cell FormatSlideshow

In [1]:

**print** "test"

test

In [17]:

**%**sx ls

Out[17]:

['derby.log',

'Getting Started with SparkSQL.ipynb',

'metastore\_db',

'output2.csv',

'Untitled.ipynb']

In [8]:

**%**sx hdfs dfs **-**ls

Out[8]:

['Found 2 items',

'-rw-r--r-- 3 bbeauchamp bbeauchamp 185038 2016-03-04 14:03 output\_file.json',

'drwxr-xr-x - bbeauchamp bbeauchamp 0 2016-03-04 14:25 test\_parquet.txt']

In [18]:

test = sqlContext.read.json("output\_file.json")

In [19]:

**print** test.take(1)

[Row(CORRIDOR=u'1701-PB340', DRIVER\_ETA\_MINS\_REM=369.0, HOUR\_OF\_DAY=19.0, KM\_REMAINING=139.03697714753346, LATITUDE=33.543056, LAT\_LON\_CORR=u'33.40\_-82.00\_33.58\_-84.52', LONGITUDE=-83.01861099999999, MINS\_REMAINING=95.0, SHIPMENT\_ID=u'304716814', carrier=u'USXI', currentLocationTime=u'2015-02-07T19:51:00.000Z', dest\_lat=33.576, dest\_lon=-84.5154, hour\_of\_day\_local=14.0, plannedArrivalTime=u'2015-02-08T02:00:00.000Z')]

In [6]:

test.printSchema()

root

|-- CORRIDOR: string (nullable = true)

|-- DRIVER\_ETA\_MINS\_REM: double (nullable = true)

|-- HOUR\_OF\_DAY: double (nullable = true)

|-- KM\_REMAINING: double (nullable = true)

|-- LATITUDE: double (nullable = true)

|-- LAT\_LON\_CORR: string (nullable = true)

|-- LONGITUDE: double (nullable = true)

|-- MINS\_REMAINING: double (nullable = true)

|-- SHIPMENT\_ID: string (nullable = true)

|-- carrier: string (nullable = true)

|-- currentLocationTime: string (nullable = true)

|-- dest\_lat: double (nullable = true)

|-- dest\_lon: double (nullable = true)

|-- hour\_of\_day\_local: double (nullable = true)

|-- plannedArrivalTime: string (nullable = true)

In [7]:

test.write.save("test\_parquet.txt", format="parquet")

In [9]:

df = sqlContext.read.parquet("test\_parquet.txt")

In [10]:

df.count()

​

Out[10]:

408

In [11]:

**print** df

DataFrame[CORRIDOR: string, DRIVER\_ETA\_MINS\_REM: double, HOUR\_OF\_DAY: double, KM\_REMAINING: double, LATITUDE: double, LAT\_LON\_CORR: string, LONGITUDE: double, MINS\_REMAINING: double, SHIPMENT\_ID: string, carrier: string, currentLocationTime: string, dest\_lat: double, dest\_lon: double, hour\_of\_day\_local: double, plannedArrivalTime: string]

In [12]:

pdf = df.toPandas()

In [13]:

type(pdf)

​

Out[13]:

pandas.core.frame.DataFrame

In [14]:

**import** pandas **as** pd

​

In [15]:

pdf.to\_csv("output2.csv", header=0)

In [ ]:

​

## Transfer a CSV file to HDFS

file name after upload via jupyter filename in HDFS home

%sx hdfs dfs -put /home/bbeauchamp/pg\_lanes\_export\_2017-05-18.csv pg\_lanes\_export\_2017-05-18.csv

# DataFrames (pyspark.sql.dataframe.DataFrame)

## Concatenate 2 pySpark DataFrames, row-wise

df1 = spark.range(3)

df1.show()

df2 = spark.range(4)

df2.show()

df = df1.union(df2)

df.show()

Output:

+---+

| id|

+---+

| 0|

| 1|

| 2|

+---+

+---+

| id|

+---+

| 0|

| 1|

| 2|

| 3|

+---+

+---+

| id|

+---+

| 0|

| 1|

| 2|

| 0|

| 1|

| 2|

| 3|

+---+

## Display DataFrame Columns and Types

df = sqlContext.sql("SELECT \* from shipment\_sums limit 5")

display(df)

Output:

DataFrame[carriers: array<string>, computedTimeUtc: string, countDeliveryLocationArrivals: bigint, countDeliveryLocationDepartures: bigint, countGeolocationUpdates: bigint, countPickupLocationArrivals: bigint, countPickupLocationDepartures: bigint, countRestatedPickupLocationArrivals: bigint, countRestatedPickupLocationDepartures: bigint, customers: array<string>, kmsDirectShipment: double, kmsTraversedShipment: double, legCount: bigint, modes: array<string>, poiFromId: bigint, poiFromLatitude: double, poiFromLongitude: double, poiFromName: string, poiToId: bigint, poiToLatitude: double, poiToLongitude: double, poiToName: string, routePoiIds: array<bigint>, routePoiNames: array<string>, secsDurationLifeCycle: double, secsMovingInTransit: double, secsStationaryInTransit: double, secsStationaryMax: double, shipmentDestinationArrivalAnalyticUtc: string, shipmentDestinationArrivalDetectedUtc: string, shipmentDestinationArrivalReportedUtc: string, shipmentDestinationDepartureAnalyticUtc: string, shipmentDestinationDepartureReportedUtc: string, shipmentEndActualUtc: string, shipmentEndLocalDate: string, shipmentEndLocalDow: bigint, shipmentEndLocalHour: bigint, shipmentEndLocalMonth: bigint, shipmentEndLocalTime: string, shipmentEndLocalTimeZone: string, shipmentEndLocalYear: bigint, shipmentId: string, shipmentOriginArrivalAnalyticUtc: string, shipmentOriginArrivalDetectedUtc: string, shipmentOriginArrivalReportedUtc: string, shipmentOriginDepartureAnalyticUtc: string, shipmentOriginDepartureDetectedUtc: string, shipmentOriginDepartureReportedUtc: string, shipmentScheduleStatus: string, shipmentStartActualUtc: string, shipmentStartLocalDate: string, shipmentStartLocalDow: bigint, shipmentStartLocalHour: bigint, shipmentStartLocalMonth: bigint, shipmentStartLocalTime: string, shipmentStartLocalTimeZone: string, shipmentStartLocalYear: bigint, shipmentSummaryId: string, shipper: string]

## DataFrame Schema

smurf\_processed\_df.printSchema()

Output:

root

|-- smfMetaData: struct (nullable = true)

| |-- defaultProcessMode: string (nullable = true)

| |-- deviceId: string (nullable = true)

| |-- endpointReceiptTime: string (nullable = true)

| |-- endpointResponseCode: string (nullable = true)

| |-- ipAddress: string (nullable = true)

## Extract a column from a pyspark DataFrame as a List

shipment\_id\_list = lane\_subset\_df.select('shipmentId').collect()

## Sample a DataFrame (approximate sample)

od\_sample = od.sample(withReplacement=False, fraction=0.04, seed=13) # Sample 4% of data

## Sort a DataFrame

od\_sample = od\_sample.sort("od\_pair", ascending=True)

Output:

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

|source\_id|dest\_id|od\_pair|

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

| 10| 1077|10-1077|

| 10| 1171|10-1171|

| 10| 136| 10-136|

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

# pyspark.sql

Select a Field from a Row

print take\_out[0]

take\_out[0].SAPShipmentID

Output:

Row(SAPShipmentID=u'0305463905')

u'0305463905'