Cheatography

PySpark Cheat Sheet

by Datacademy.ai (Datacademy.ai) via cheatography.com/174553/cs/36684/

What is Pyspark?

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed enviro-nment.

Initializing SparkSession

```
>>> from pyspark.sql import SparkSession
>>> spark = SparkS ession \
                    .bu ilder \
                    .ap pNa me( " - Python Spark SQL basic exampl e" ) \
                    .co nfi g("s - par k.s ome.co nfi g.o pti - on", " som e-v alu e") \
                    .ge tOr Cre - ate()
```

A SparkSession can be used create DataFrame, register
DataFrame as tables, execute SQL over tables, cache
tables, and read parquet files

Creating DataFrames

From RDDs

```
>>> from pyspar k.s ql.t ypes import *
```

Creating DataFrames from RDDs

Infer Schema

```
>>> sc = spark.s pa rkC ontext
>>> lines = sc.tex tFi le( " - peo ple.tx t")
>>> parts = lines.m ap (lambda l: l.spli t(", "))
>>> people = parts.m ap (lambda p: Row(na me= p[0 ],a ge= int - (p[ 1])))
>>> peopledf = spark.c re ate - Dat aFr ame (pe ople)
```

Creating DataFrames from RDDs

Specify Schema

```
>>> people = parts.m ap (lambda p: Row(na me= p[0], age=in t(p - [1].st rip ()))
>>> schema String = "name age"
>>> fields = [Struc tFi eld - (fi eld _name, String Type(), True) for field_name in schema - Str ing.sp lit()]
>>> schema = Struct Typ e(f - ields)
>>> spark.c re ate Dat aFr ame - (pe ople, schema ).s how()
```

From Spark Data Sources

JSON

```
>>> df = spark.r ea d.j son - ("cu sto mer.js on")
>>> df.show()
>>> df2 = spark.r ea d.l oad - ("pe opl e.j son ", format - ="js on")
```

From Spark Data Sources

Parquet Files

```
>>> df3 = spark.r ea d.l oad - ("us ers.pa rqu et")
```

From Spark Data Sources

Text Files

```
>>> df4 = spark.r ea d.t ext - ("pe opl e.t xt")
```

Duplicate Values

```
>>> df = df.dropDuplicates()
```

Inspect Data

`>>> df.dtypes`	Return df column names and data types
`>>> df.show()`	Display the content of df
`>>> df.head()`	Return first n rows
`>>> df.first()`	Return first row
`>>> df.take(2)`	Return the first n rows
`>>> df.schema`	Return the schema of df
`>>> df.des - cri be( ).s - how()`	Compute summary statistics
`>>> df.columns`	Return the columns of df
`>>> df.count()`	Count the number of rows in df

Inspect Data (cont)

`>>> df.dis - tin ct( ).c - ount()`	Count the number of distinct rows in df
`>>> df.pri - ntS chema()`	Print the schema of df
`>>> df.exp - lain()`	Print the (logical and physical) plans

Queries

```
>>> from pyspark.sql import functions as F
```

Select- Query

```
>>> df.select("firstName").show()
>>> df.sel ect ("fi rst Nam - e","l ast Nam e").s how()
>>> df.sel ect (df ["fi rst - Nam e"], df[ " age "]+ 1).show()
>>> df.sel ect (df ['age'] > 24).show()
```

Between- Query

```
>>> df.select(df.age.between(22, 24)).show()
```

Substring- Query

```
df.select(df.firstName.substr(1, 3) \
                    .al ias - ("na me") ).c oll ect()
```

Startswith, Endswith- Query

```
>>> df.select("firstName",
        df.l as tNa me.s ta - rts wit h("S m")).show()
>>> df.sel ect (df.la stN - ame.en dsw ith ("th " )).s how(
```

Like - Query

```
>>> df.select("firstName",
        df.l as tNa me.l ik - e("S mit h")).show()
```

When- Query

```
>>> df.select("firstName",
                    F.w hen - (df.age > 30, 1) \
                  .ot her wis - e(0)) \
                  .show()
>>> df[df.f ir stN ame.is - in( " Jan e","B ori s")].co - llect()
```

GroupBy

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

Filter

```
>>> df.filter(df["age"]>24).show()
```

Add Columns

```
>>> df = df.withColumn('city',df.address.ci \
        .wi thC olu mn( 'po sta lCo de' ,df.ad dre ss.p os tal Code) \
        .wi thC olu mn( 'st ate ',d f.a ddr ess.state) \
        .wi thC olu mn( 'st ree tAd dre ss' ,df.ad dre ss.s tr eet Add ress) \
        .wi thC olu mn( 'te leP hon eNu mber',
         exp lod e(d f.p hon eNu mb r.n umber)) \
        .wi thC olu mn( 'te leP hon eType',
         exp lod e(d f.p hon eNu mb r.t ype))
```

Update Columns

```
>>> df = df.withColumnRenamed('telePhoneNum \
                    'ph one Num ber')
```

Remove Columns

```
>>> df = df.drop("address", "phoneNumber")
>>> df = df.dro p(d f.a ddr - ess ).d rop (df.ph one Number)
```

Sort

```
>>> peopledf.sort(peopledf.age.desc())
>>> df.sor t("a ge", ascend ing =F lse ).c oll ect()
>>> df.ord erB y([ " age " ,"ci ty cen din g=[ 0,1 ]).c ol lect()
```

Missing & Replacing Values

```
>>> df.na.fill(50).show()
>>> df.na.d ro p().show()
>>> df.na.r ep lac e(10,
20).show()
```

Repartitioning

```
>>>
df.repartition(10).rdd.getNumPartitions()
>>> df.coa les ce( 1).r dd.ge tNu mPa -
rti tions()
```

Registering DataFrames as Views

```
>>>
peopledf.createGlobalTempView("peoplet y")
>>> df.cre ate Tem pVi ew( " cus tom
er")
>>> df.cre ate OrR epl ace Tem pVi -
ew( " cus tom er")
```

Query Views

```
>>> df5 = spark.sql("SELECT *
FROM customer").show()
>>> peopledf2 =spark.sq l("S -
ELECT *FROM global _te mp.p eo -
ple ") \
.show()
```

Output- Data Structures

```
>>> rdd1 = df.rdd
```

Convert df into an RDD

```
>>> df.toJ SON ().f irst()
```

Convert df into a RDD of string

```
>>> df.toP andas()
```

Return the contents of df as Pandas DataFrame

Output- Write & Save to Files

```
>>> df.select("firstName",
"city")\
.write.sa ve( " nam eAn dCi -
y.p ar que t")
>>> df.sel ect ("fi rst Nam e",
" age ") \
.write.sa ve( " nam esA ndA -
ges.js on", for mat ="js on")
```

Stopping SparkSession

```
>>> spark.stop()
```