

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 environment.

Initializing SparkSession

```
>>> from pyspark.sql import
SparkSession
>>> spark = SparkSession \
    .builder \
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

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

Creating DataFrames

From RDDs

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

Creating DataFrames from RDDs

Infer Schema

```
>>> sc = spark.sparkContext
>>> lines = sc.textFile("people.txt")
>>> parts = lines.map(lambda l: l.split(", "))
>>> people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))
>>> peopledf = spark.createDataFrame(people)
```

Creating DataFrames from RDDs

Specify Schema

```
>>> people = parts.map(lambda p: Row(name=p[0], age=int(p[1].strip())))
>>> schemaString = "name age"
>>> fields = [StructField(field_name, StringType, True) for field_name in schemaString.split()]
>>> schema = StructType(fields)
>>> spark.createDataFrame(people, schema).show()
```

From Spark Data Sources

JSON

```
>>> df = spark.read.json("customer.json")
>>> df.show()
>>> df2 = spark.read.load("people.json", format="json")
```

From Spark Data Sources

Parquet Files

```
>>> df3 = spark.read.load("users.parquet")
```

From Spark Data Sources

Text Files

```
>>> df4 = spark.read.text("people.txt")
```

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.describe().show() Compute summary statistics
>>> df.columns    Return the columns of df
>>> df.count()    Count the number of rows in df
```

Inspect Data (cont)

```
>>> df.distinct().count()
df.printSchema()
df.explain()
```

Count the number of distinct rows in df

Print the schema of df

Print the (logical and physical) plans

Queries

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

Select- Query

```
>>> df.select("firstName").show()
>>> df.select("firstName", "lastName").show()
>>> df.select(df["firstName"], df["age"] + 1).show()
>>> df.select(df['age'] > 24).show()
```

Between- Query

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

Substring- Query

```
df.select(df.firstName.substr(1, 3))
df.alias("name").collect()
```

Startswith, Endswith- Query

```
>>> df.select("firstName", "lastName").show()
df.select(df.lastName.endswith("th")).show()
```

Like - Query

```
>>> df.select("firstName", "lastName").show()
df.select(df.lastName.like("Smith")).show()
```

When- Query

```
>>> df.select("firstName", "age").show()
df.filter(df.age > 30).show()
df.filter(df.age > 30).count().show()
df.filter(df.age > 30).collect()
```

GroupBy

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

Filter

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

Add Columns

```
>>> df = df.withColumn('city', df.address.city)
df.withColumn('postalCode', df.address.postalCode)
df.withColumn('state', df.address.state)
df.withColumn('streetAddress', df.address.streetAddress)
df.withColumn('telephoneNumber', df.address.phoneNumber)
df.withColumn('telemarketingType', df.address.marketingType)
df.withColumn('phoneNumber', df.phoneNumber)
```

Update Columns

```
>>> df = df.withColumnRenamed('telephoneNumber', 'phoneNumber')
```

Remove Columns

```
>>> df = df.drop("address", "phoneNumber")
df.drop(df.address, df.phoneNumber)
```

Sort

```
>>> peopleDF.sort(peopleDF.age.desc())
df.sort("age", ascending=False).collect()
df.orderBy("age", "city").collect()
```

Missing & Replacing Values

```
>>> df.na.fill(50).show()
>>> df.na.drop().show()
>>> df.na.replace(10,
20).show()
```

Repartitioning

```
>>>
df.repartition(10).rdd.getNumPartitions()
>>> df.coalesce(1).rdd.getNumPartitions()
```

Registering DataFrames as Views

```
>>> peopledf.createGlobalTempView("people")
>>> df.createTempView("customer")
>>> df.createOrReplaceTempView("customer")
```

Query Views

```
>>> df5 = spark.sql("SELECT *
FROM customer").show()
>>> peopledf2 = spark.sql("SELECT *FROM global _temp.people") \
.show()
```

Output- Data Structures

```
>>> rdd1 = df.rdd
| Convert df into an RDD
>>> df.toJSON().first()
| Convert df into a RDD of string
>>> df.toPandas()
| Return the contents of df as Pandas DataFrame
```

Output- Write & Save to Files

```
>>> df.select("firstName",
"city") \
.write.save("nameAndCity.parquet")
>>> df.select("firstName",
"age") \
.write.save("namesAndAges.json", format="json")
```

Stopping SparkSession

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



By **Datacademy.ai**
(Datacademy.ai)

cheatography.com/datacademy-ai/

Not published yet.

Last updated 24th January, 2023.

Page 3 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>