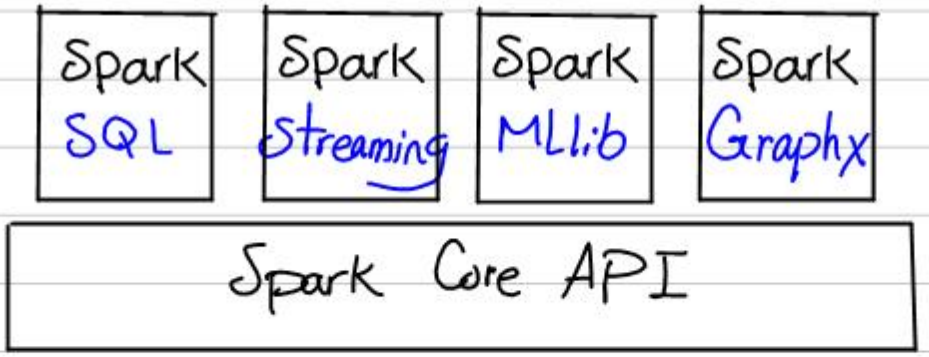
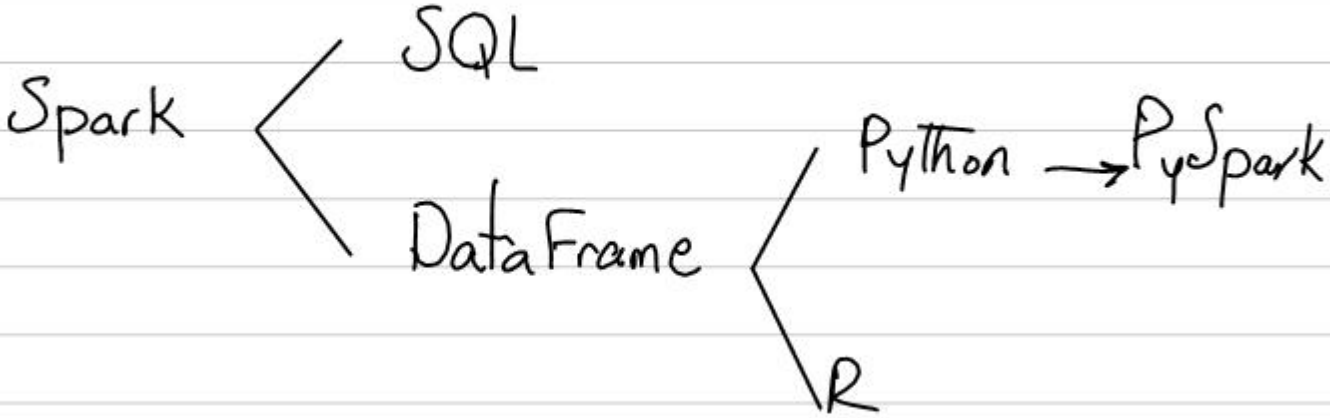
Apache Spark | PySpark

- is developed on Scala



- Task Schedoling
- Memory Management
- Interacting with Storage System
- Lazy evaluation



Spark

vs.

Pandas

Tabular data

Can handle millions rows

limited to a single machine

↓Spark
distributed

Hadoop

distributed

Compute System
(MapReduce)
Storage System
(HDFS)
Hadoop Distributed File System

↓

read from & write to disk

↓Spark

in memory

Spark Data APIs
- DataFrames → high_level API
- RDDs → low_level API

```
From pyspark.sql.functions import min, max, avg, asc, desc
From pyspark.sql.types import FloatType, IntegerType
From pyspark.ml.feature import Imputer
From pyspark.ml.feature import VectorAssembler
From pyspark.ml.regression import LinearRegression
From pyspark.ml.classification import LogesticRegression,
                                      RandomForestClassifier
```

Pandas	PySpark
import pandas as pd	import pyspark From pyspark.sql import SparkSession spark = SparkSession.builder.getOrCreate()
df = pd.read_csv(filepath, header=	df = spark.read.csv(filepath, header=)
pd.to_csv(' ')	df.write.format('csv').save(' ')
df.head() df.sample(5) df.tail() df.describe()	df.show() df.head() df.take() df.limit() df.collect() ← Be careful! df.describe().show()
df.columns	df.columns
df.shape	len(df.columns) → # df.count() → #
df.dtypes	df.dtypes df.printSchema()
df['Col name']	df.select('Col name')

Pandas	PySpark
df['new col'] = value	df.withColumn('newcol', value)
df.drop('col name')	df.drop('col name')
df.rename({'old':'new'})	df.withColumnRename('old name', 'new name')
df[df['Age']>35]	df[df['Age']>35] df.filter('Age > 35') *df.where('Age < 30')
df.toPandas(SparkDf)	spark.createDataFrame(pd_df)

df	Age	Sex	weight
1	18	M	72
2	22	F	61
3	22	F	57
4	24	F	63
5	18	M	78
6	18	F	60
	⋮	⋮	⋮

```
df.groupBy('Age').count() | mean('weight')
```

Age	count	mean(w)	min(w)
18	28	60	42
21	13	61	43
22	11	63	48
24	18	67	49
	⋮		

↑ascending صعودی (کوچک به بزرگ)

↓descending نزولی (بزرگ به کوچک)