Introduction to Big Data and Data Science (CSCE 5300)*

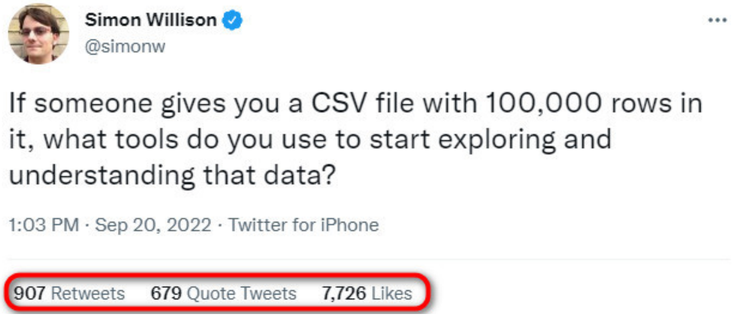
Yunhe Feng

Assistant Professor, Department of Computer Science and Engineering

12th September, 2024

*The teaching materials are reorganized and reformed based on Prof. Ravi Vadapalli's slides (Ravi.Vadapalli@unt.edu, UNT & University of Miami)

How to Process a CSV File with 100,000 Rows



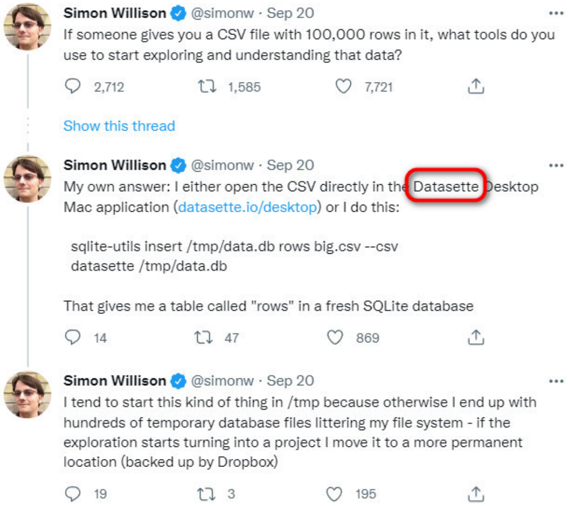
Simon Willison
@simonw

If someone gives you a CSV file with 100,000 rows in it, what tools do you use to start exploring and understanding that data?

1:03 PM · Sep 20, 2022 · Twitter for iPhone

907 Retweets 679 Quote Tweets 7,726 Likes

How to Process a CSV File with 100,000 Rows


Simon Willison @simonw · Sep 20
If someone gives you a CSV file with 100,000 rows in it, what tools do you use to start exploring and understanding that data?
2,712 1,585 7,721

Show this thread

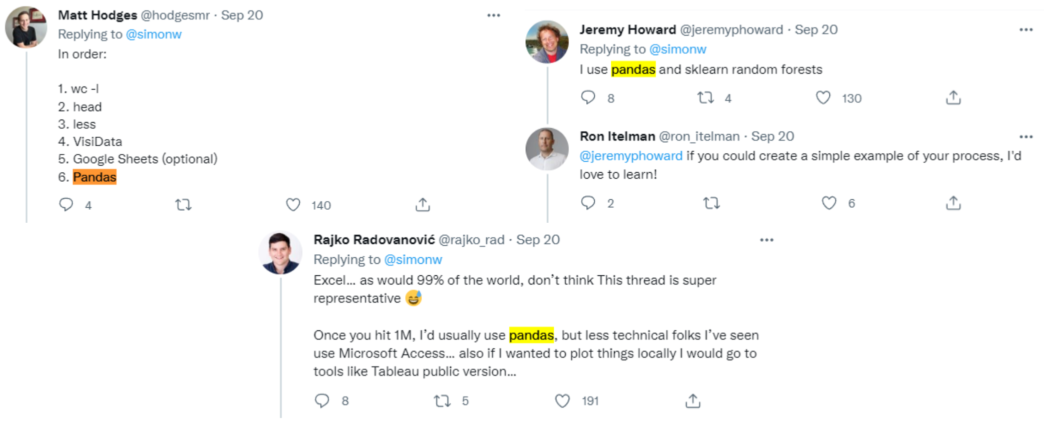
Simon Willison @simonw · Sep 20
My own answer: I either open the CSV directly in the Datasette Desktop Mac application (datasette.io/desktop) or I do this:

```
sqlite-utils insert /tmp/data.db rows big.csv --csv
datasette /tmp/data.db
```

That gives me a table called "rows" in a fresh SQLite database
14 47 869

Simon Willison @simonw · Sep 20
I tend to start this kind of thing in /tmp because otherwise I end up with hundreds of temporary database files littering my file system - if the exploration starts turning into a project I move it to a more permanent location (backed up by Dropbox)
19 3 195

How to Process a CSV File with 100,000 Rows



Matt Hodges @hodgesmr · Sep 20
Replying to @simonw
In order:

1. wc -l
2. head
3. less
4. VisiData
5. Google Sheets (optional)
6. Pandas

4 140

Jeremy Howard @jeremyphoward · Sep 20
Replying to @simonw
I use pandas and sklearn random forests

8 4 130

Ron Itelman @ron_itelman · Sep 20
@jeremyphoward if you could create a simple example of your process, I'd love to learn!

2 6

Rajko Radovanović @rajko_rad · Sep 20
Replying to @simonw
Excel... as would 99% of the world, don't think This thread is super representative 😅

Once you hit 1M, I'd usually use pandas, but less technical folks I've seen use Microsoft Access... also if I wanted to plot things locally I would go to tools like Tableau public version...

8 5 191

What is pandas

What is pandas

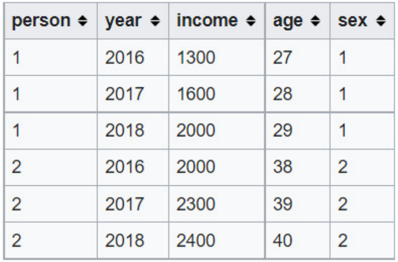
pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

pandas VS NumPy

- pandas is built on the NumPy library and written in languages like Python, Cython, and C. It is a convenience wrapper over NumPy.
- NumPy is a Python library used for working with arrays.

Why is pandas Called pandas

- **pandas**: The library's name derives from **pan**el **da**ta, a common term for multidimensional data sets encountered in statistics and econometrics[1].
- **Panel Data**: In statistics and econometrics, panel data is multi-dimensional data involving measurements over time. Panel data is a subset of longitudinal data where observations are for the same subjects each time. – Wikipedia



person	year	income	age	sex
1	2016	1300	27	1
1	2017	1600	28	1
1	2018	2000	29	1
2	2016	2000	38	2
2	2017	2300	39	2
2	2018	2400	40	2

[1]"pandas: a Foundational Python Library for Data Analysis and Statistics" by Wes McKinney

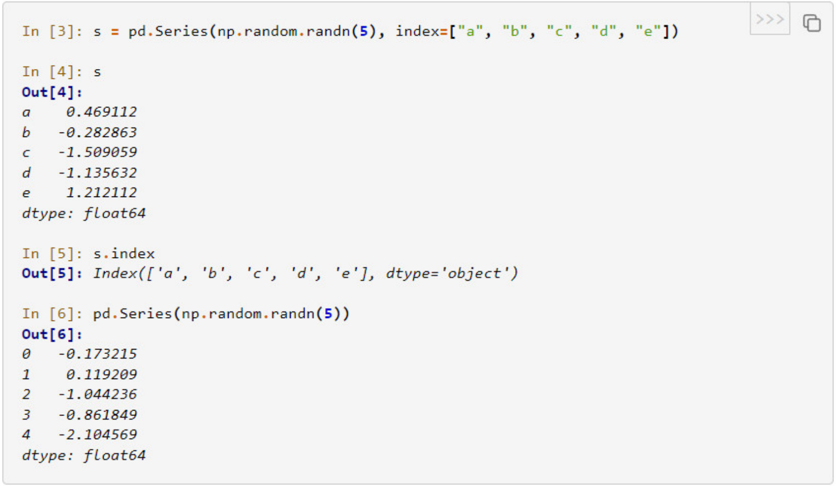
pandas VS PySpark

- pandas run operations on a single machine whereas PySpark runs on multiple machines.
- The DataFrame structure in PySpark is conceptually similar to (and inspired by) the one in pandas.

pandas Data Structures

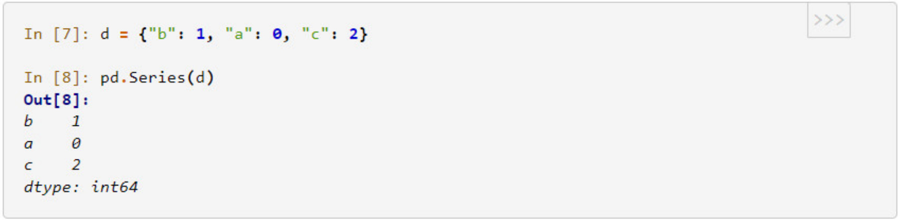
- Series: a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index.
- DataFrame: a 2-dimensional labeled data structure with columns of potentially different types.

Series: From ndarray



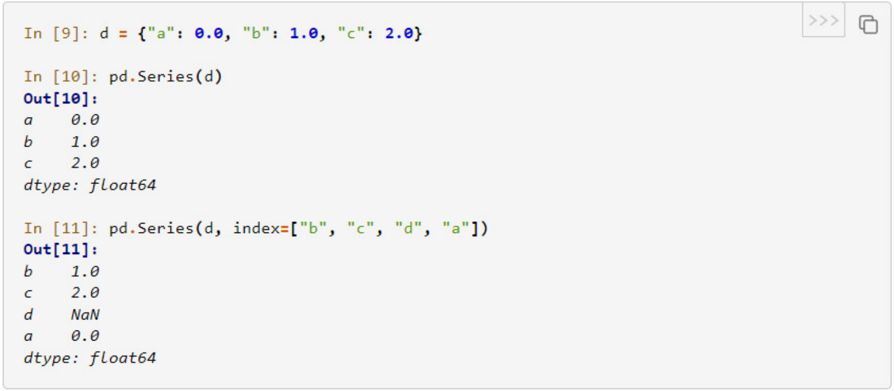
```
In [3]: s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])

In [4]: s
Out[4]:
a    0.469112
b   -0.282863
c   -1.509059
d   -1.135632
e    1.212112
dtype: float64

In [5]: s.index
Out[5]: Index(['a', 'b', 'c', 'd', 'e'], dtype='object')

In [6]: pd.Series(np.random.randn(5))
Out[6]:
0   -0.173215
1    0.119209
2   -1.044236
3   -0.861849
4   -2.104569
dtype: float64
```

Series: From dict



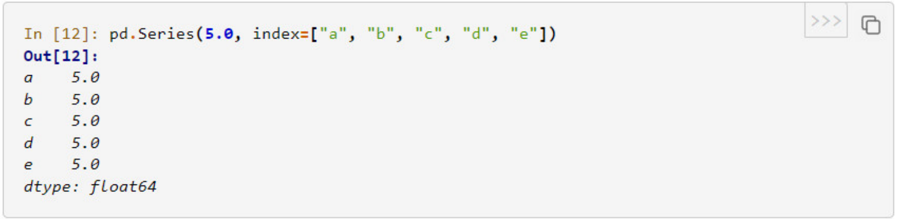
```
In [7]: d = {"b": 1, "a": 0, "c": 2}

In [8]: pd.Series(d)
Out[8]:
b    1
a    0
c    2
dtype: int64
```

Series: From dict



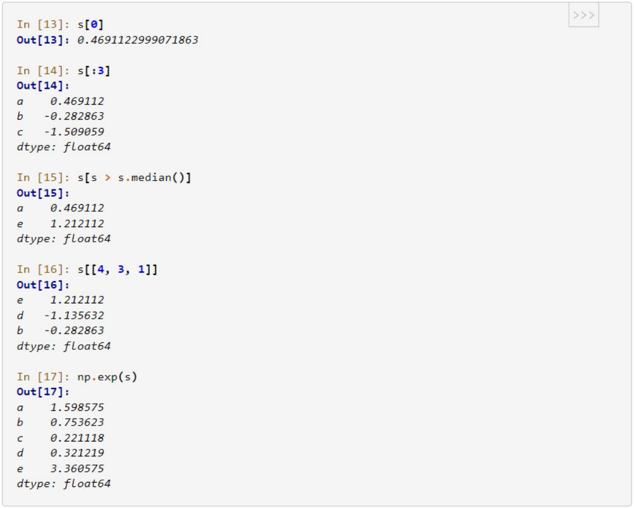
```
In [9]: d = {"a": 0.0, "b": 1.0, "c": 2.0}

In [10]: pd.Series(d)
Out[10]:
a    0.0
b    1.0
c    2.0
dtype: float64

In [11]: pd.Series(d, index=["b", "c", "d", "a"])
Out[11]:
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Series: From scalar value



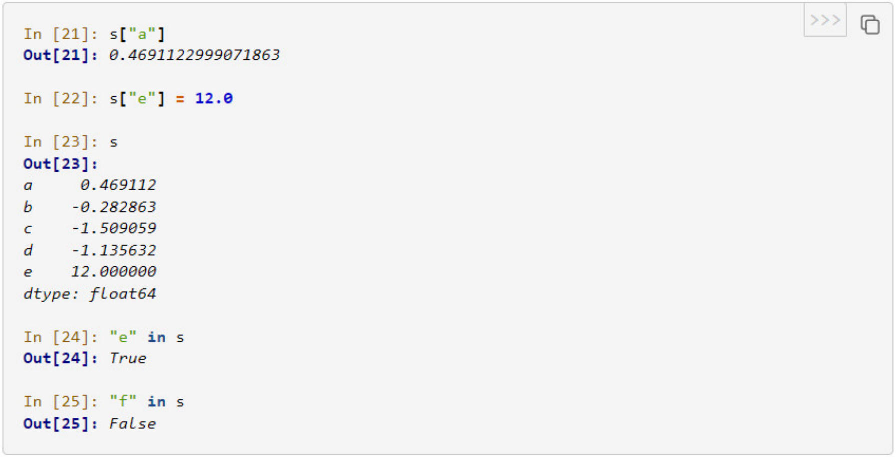
```
In [12]: pd.Series(5.0, index=["a", "b", "c", "d", "e"])
Out[12]:
a    5.0
b    5.0
c    5.0
d    5.0
e    5.0
dtype: float64
```

Series is ndarray-like



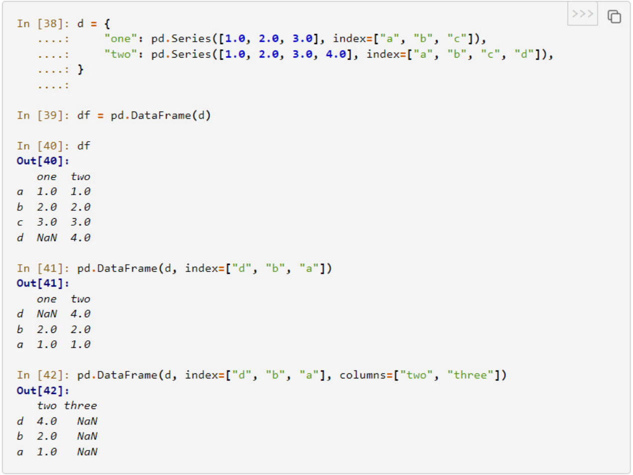
```
In [13]: s[0]
Out[13]: 0.4691122999071863

In [14]: s[:3]
Out[14]:
a    0.469112
b   -0.282863
c   -1.509059
dtype: float64

In [15]: s[s > s.median()]
Out[15]:
a    0.469112
e    1.212112
dtype: float64

In [16]: s[[4, 3, 1]]
Out[16]:
e    1.212112
d   -1.135632
b   -0.282863
dtype: float64

In [17]: np.exp(s)
Out[17]:
a    1.598575
b    0.753623
c    0.221118
d    0.321219
e    3.360575
dtype: float64
```

Series is dict-like



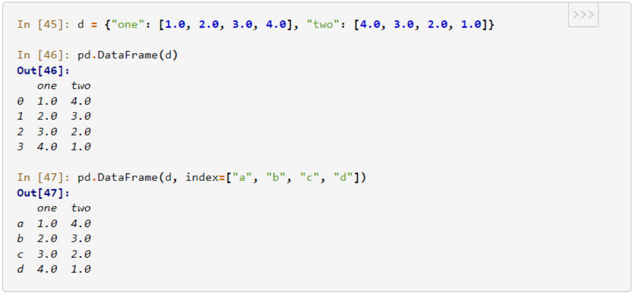
```
In [21]: s["a"]
Out[21]: 0.4691122999071863

In [22]: s["e"] = 12.0

In [23]: s
Out[23]:
a     0.469112
b    -0.282863
c    -1.509059
d    -1.135632
e    12.000000
dtype: float64

In [24]: "e" in s
Out[24]: True

In [25]: "f" in s
Out[25]: False
```

DataFrame: From dict of Series or dicts



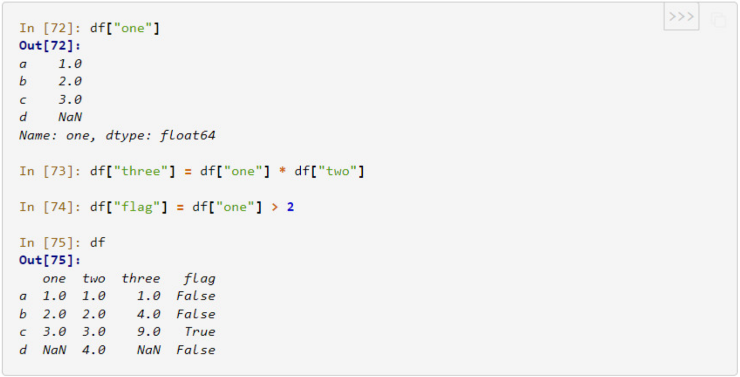
```
In [38]: d = {
   ....:     "one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]),
   ....:     "two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),
   ....: }
   ....:

In [39]: df = pd.DataFrame(d)

In [40]: df
Out[40]:
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0

In [41]: pd.DataFrame(d, index=["d", "b", "a"])
Out[41]:
   one  two
d  NaN  4.0
b  2.0  2.0
a  1.0  1.0

In [42]: pd.DataFrame(d, index=["d", "b", "a"], columns=["two", "three"])
Out[42]:
   two three
d  4.0   NaN
b  2.0   NaN
a  1.0   NaN
```

DataFrame: From dict of ndarrays / lists



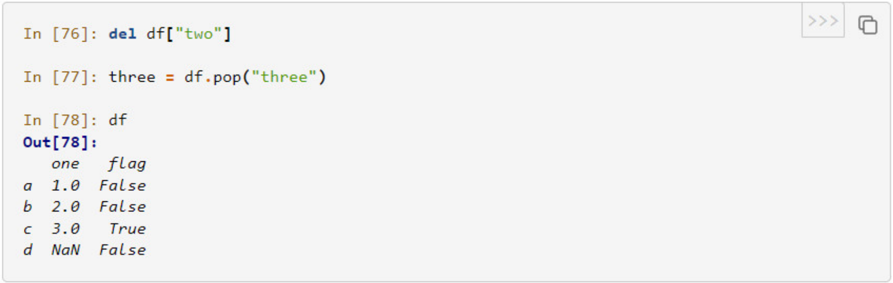
```
In [45]: d = {"one": [1.0, 2.0, 3.0, 4.0], "two": [4.0, 3.0, 2.0, 1.0]}

In [46]: pd.DataFrame(d)
Out[46]:
   one  two
0  1.0  4.0
1  2.0  3.0
2  3.0  2.0
3  4.0  1.0

In [47]: pd.DataFrame(d, index=["a", "b", "c", "d"])
Out[47]:
   one  two
a  1.0  4.0
b  2.0  3.0
c  3.0  2.0
d  4.0  1.0
```

DataFrame: Column Selection



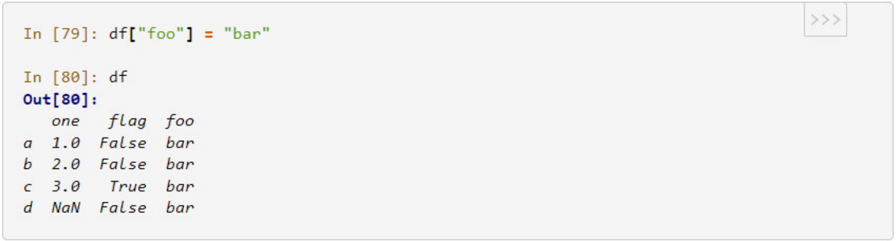
```
In [72]: df["one"]
Out[72]:
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64

In [73]: df["three"] = df["one"] * df["two"]

In [74]: df["flag"] = df["one"] > 2

In [75]: df
Out[75]:
   one  two  three   flag
a  1.0  1.0    1.0  False
b  2.0  2.0    4.0  False
c  3.0  3.0    9.0   True
d  NaN  4.0    NaN  False
```

DataFrame: Column Deletion



```
In [76]: del df["two"]

In [77]: three = df.pop("three")

In [78]: df
Out[78]:
   one   flag
a  1.0  False
b  2.0  False
c  3.0   True
d  NaN  False
```

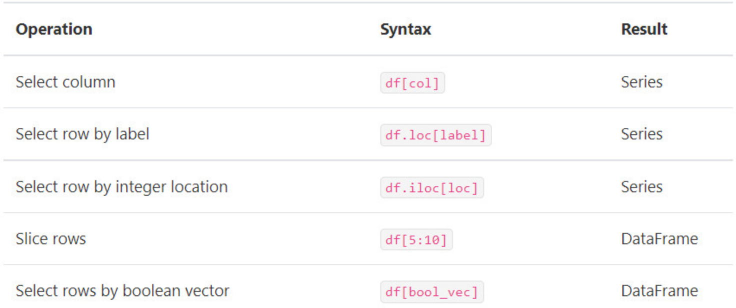
DataFrame: Column Addition



```
In [79]: df["foo"] = "bar"

In [80]: df
Out[80]:
   one   flag  foo
a  1.0  False  bar
b  2.0  False  bar
c  3.0   True  bar
d  NaN  False  bar
```

*The teaching materials are reorganized and reformed based on Prof. Ravi Vadapalli's slides (Ravi.Vadapalli@unt.edu, UNT & University of Miami)

DataFrame: Indexing / Selection



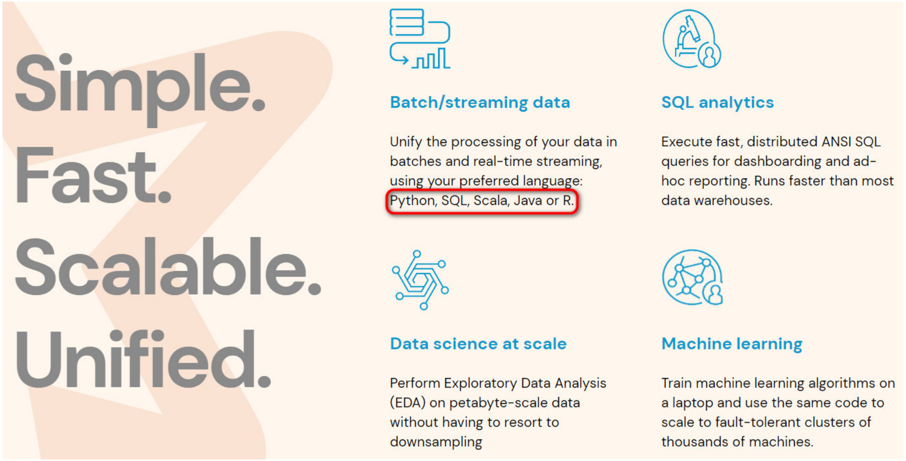
Operation	Syntax	Result
Select column	`df[col]`	Series
Select row by label	`df.loc[label]`	Series
Select row by integer location	`df.iloc[loc]`	Series
Slice rows	`df[5:10]`	DataFrame
Select rows by boolean vector	`df[bool_vec]`	DataFrame

What is PySpark

What is Spark

Apache Spark is a data processing framework that can quickly perform processing tasks on very large data sets, and can also distribute data processing tasks across multiple computers, either on its own or in tandem with other distributed computing tools.

What is Spark



Simple.
Fast.
Scalable.
Unified.

Batch/streaming data

Unify the processing of your data in batches and real-time streaming, using your preferred language: Python, SQL, Scala, Java or R.

SQL analytics

Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting. Runs faster than most data warehouses.

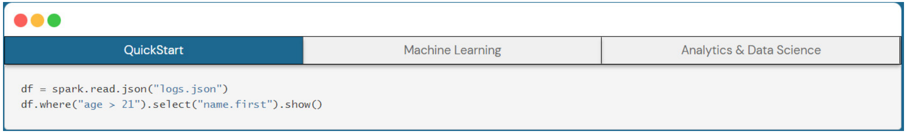
Data science at scale

Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling

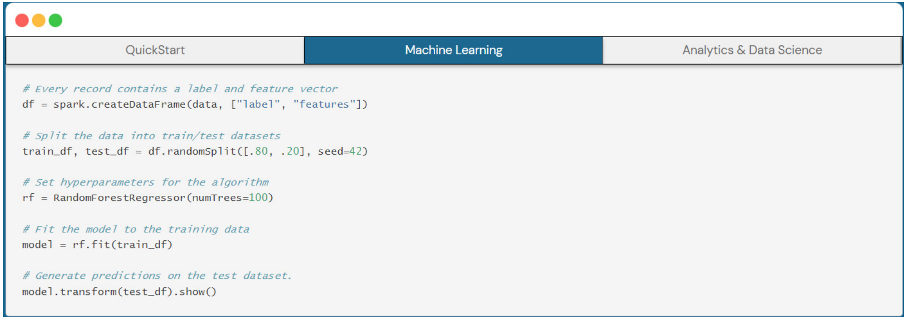
Machine learning

Train machine learning algorithms on a laptop and use the same code to scale to fault-tolerant clusters of thousands of machines.

What is Spark

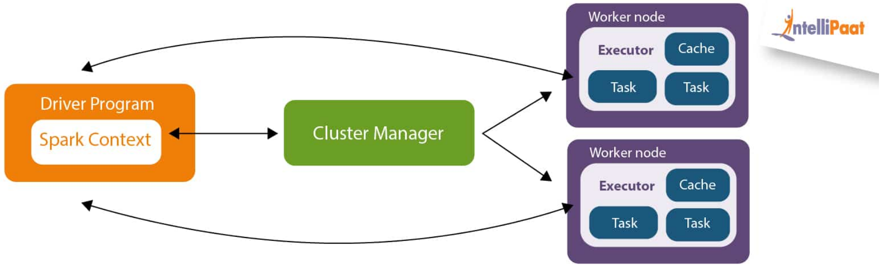


```
df = spark.read.json("logs.json")
df.where("age > 21").select("name.first").show()
```



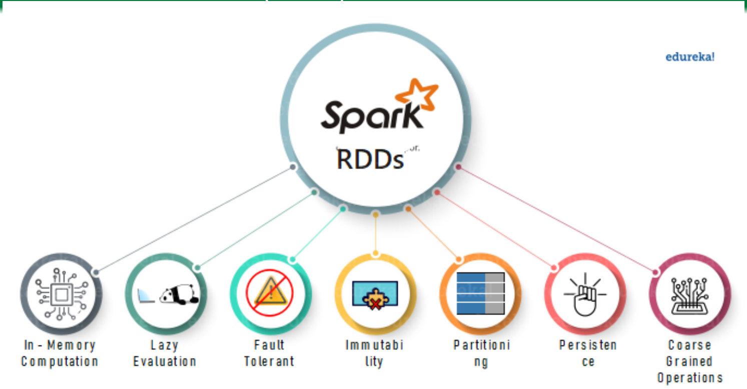
```
# Every record contains a label and feature vector
df = spark.createDataFrame(data, ["label", "features"])

# Split the data into train/test datasets
train_df, test_df = df.randomSplit([.80, .20], seed=42)

# Set hyperparameters for the algorithm
rf = RandomForestRegressor(numTrees=100)

# Fit the model to the training data
model = rf.fit(train_df)

# Generate predictions on the test dataset.
model.transform(test_df).show()
```

Spark Architecture



- Driver Program: create SparkContext and translate the user-written code into jobs
- Cluster Manager: allocate resource and split a job into multiple smaller tasks
- Worker Nodes: execute the tasks

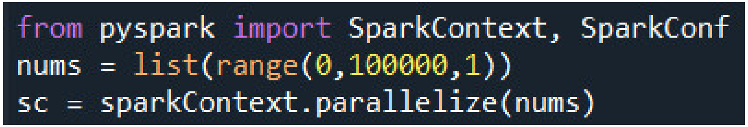
Spark Resilient Distributed Dataset (RDD)



- RDD: a programming abstraction that represents an immutable collection of objects that can be split across a computing cluster
- Operations on RDDs: can also be split across the cluster and executed in a parallel batch process

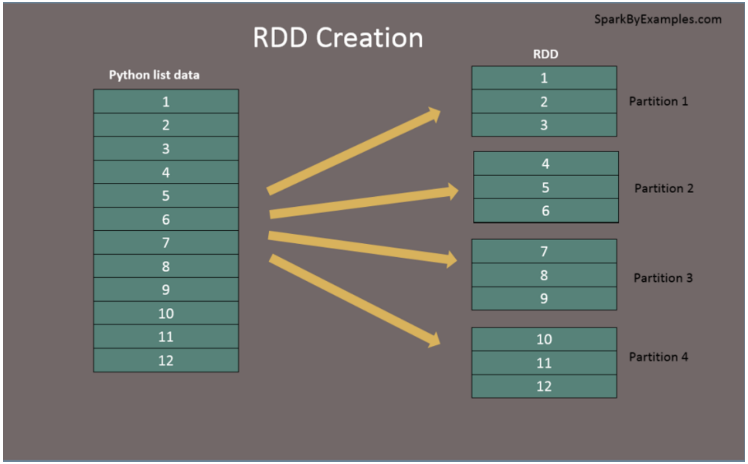
RDD: Parallelize()

- Parallelize() is a function in SparkContext
- Create Resilient Distributed Datasets (RDD) from a list collection
- Each dataset in RDD is divided into logical partitions, each partition may be computed on a different node of the cluster

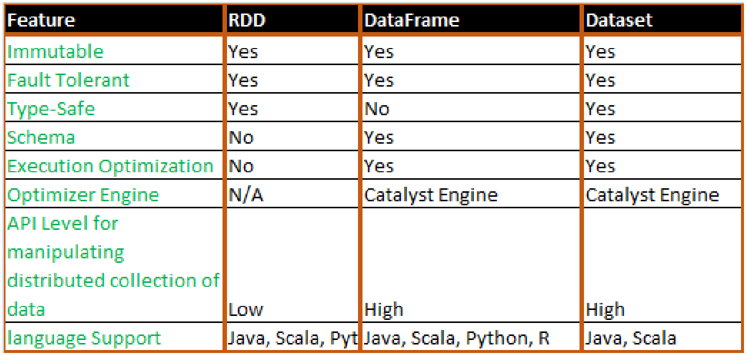


```
from pyspark import SparkContext, SparkConf
nums = list(range(0,100000,1))
sc = sparkContext.parallelize(nums)
```

RDD: Parallelize()



RDD VS DataFrame

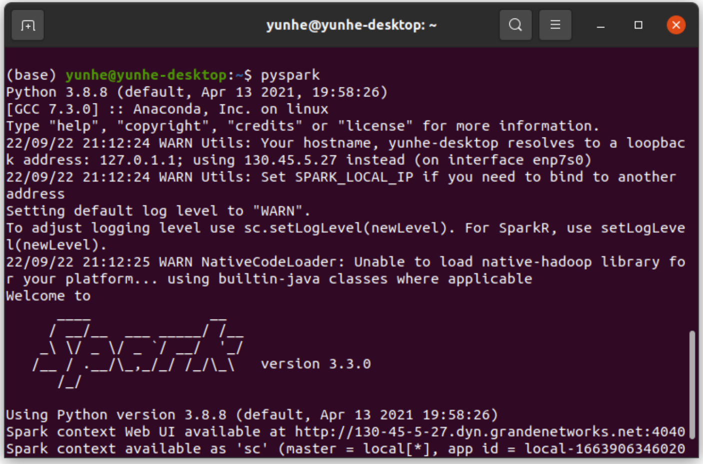


Feature	RDD	DataFrame	Dataset
Immutable	Yes	Yes	Yes
Fault Tolerant	Yes	Yes	Yes
Type-Safe	Yes	No	Yes
Schema	No	Yes	Yes
Execution Optimization	No	Yes	Yes
Optimizer Engine	N/A	Catalyst Engine	Catalyst Engine
API Level for manipulating distributed collection of data	Low	High	High
language Support	Java, Scala, Pyt	Java, Scala, Python, R	Java, Scala

Install PySpark on Linux

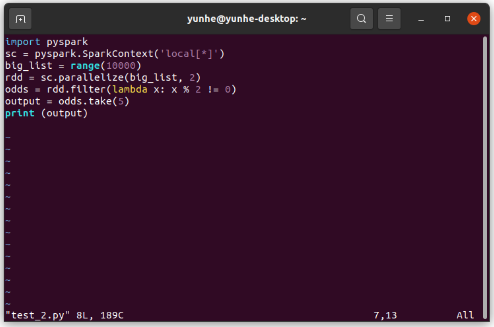
- https://sparkbyexamples.com/pyspark/install-pyspark-in-anaconda-jupyter-notebook/
- https://spark.apache.org/docs/latest/api/python/getting_started/install.html
- https://zhangdijohn.medium.com/pyspark-3-ubuntu-20-04-installation-2792c5c221de

Install PySpark on Linux

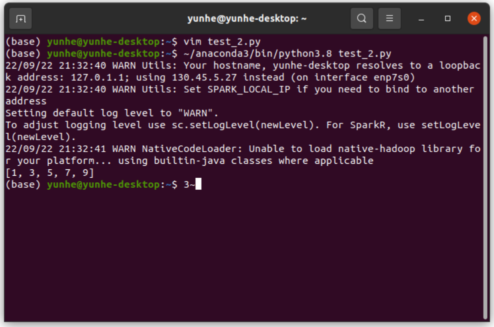

yunhe@yunhe-desktop: ~

```
(base) yunhe@yunhe-desktop:~$ pyspark
Python 3.8.8 (default, Apr 13 2021, 19:58:26)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
22/09/22 21:12:24 WARN Utils: Your hostname, yunhe-desktop resolves to a loopbac
k address: 127.0.1.1; using 130.45.5.27 instead (on interface enp7s0)
22/09/22 21:12:24 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
22/09/22 21:12:25 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.3.0
      /_/

Using Python version 3.8.8 (default, Apr 13 2021 19:58:26)
Spark context Web UI available at http://130-45-5-27.dyn.grandenetworks.net:4040
Spark context available as 'sc' (master = local[*], app id = local-1663906346020
```

PySpark Coding Examples

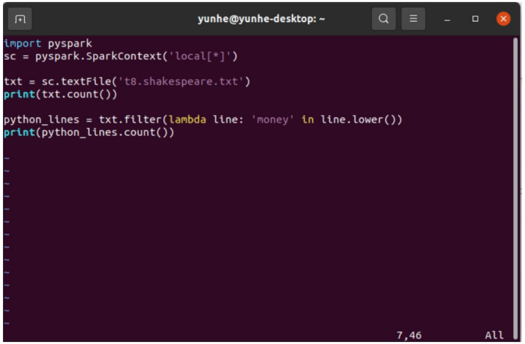


```
import pyspark
sc = pyspark.SparkContext('local[*]')
big_list = range(10000)
rdd = sc.parallelize(big_list, 2)
odds = rdd.filter(lambda x: x % 2 != 0)
output = odds.take(5)
print (output)

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"test_2.py" 8L, 189C                                    7,13          All
```

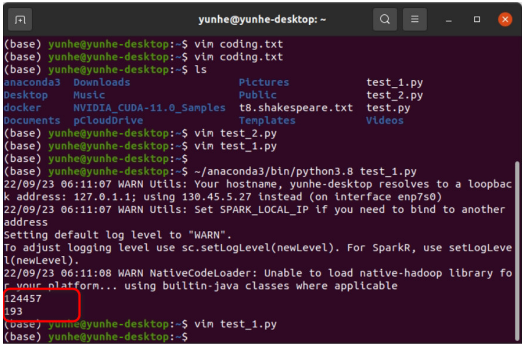


```
(base) yunhe@yunhe-desktop:~$ vim test_2.py
(base) yunhe@yunhe-desktop:~$ ~/anaconda3/bin/python3.8 test_2.py
22/09/22 21:32:40 WARN Utils: Your hostname, yunhe-desktop resolves to a loopbac
k address: 127.0.1.1; using 130.45.5.27 instead (on interface enp7s0)
22/09/22 21:32:40 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
22/09/22 21:32:41 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
[1, 3, 5, 7, 9]
(base) yunhe@yunhe-desktop:~$ 3~
```

PySpark Coding Examples



```
import pyspark
sc = pyspark.SparkContext('local[*]')

txt = sc.textFile('t8.shakespeare.txt')
print(txt.count())

python_lines = txt.filter(lambda line: 'money' in line.lower())
print(python_lines.count())
```



```
(base) yunhe@yunhe-desktop:~$ vim coding.txt
(base) yunhe@yunhe-desktop:~$ vim coding.txt
(base) yunhe@yunhe-desktop:~$ ls
anaconda3  Downloads                  Pictures             test_1.py
Desktop    Music                      Public               test_2.py
docker     NVIDIA_CUDA-11.0_Samples   t8.shakespeare.txt   test.py
Documents  pCloudDrive                Templates            Videos
(base) yunhe@yunhe-desktop:~$ vim test_2.py
(base) yunhe@yunhe-desktop:~$ vim test_1.py
(base) yunhe@yunhe-desktop:~$
(base) yunhe@yunhe-desktop:~$ ~/anaconda3/bin/python3.8 test_1.py
22/09/23 06:11:07 WARN Utils: Your hostname, yunhe-desktop resolves to a loopbac
k address: 127.0.1.1; using 130.45.5.27 instead (on interface enp7s0)
22/09/23 06:11:07 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
22/09/23 06:11:08 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
124457
193
(base) yunhe@yunhe-desktop:~$ vim test_1.py
(base) yunhe@yunhe-desktop:~$
```

Assignment-3 (4.0 pts.)

- DataFrames (2 pts.)
- Install PySpark (1 pt.)
- Run PySpark scripts (1 pt.)