Merging, Cleaning and Shaping Data

Agenda

Becoming a Data Ninja

- Merge & joins
- Concat
- Clean

Group Exercise: European Trade

Big Data Processing with Spark

- Databricks CE

Agenda

Becoming a Data Ninja

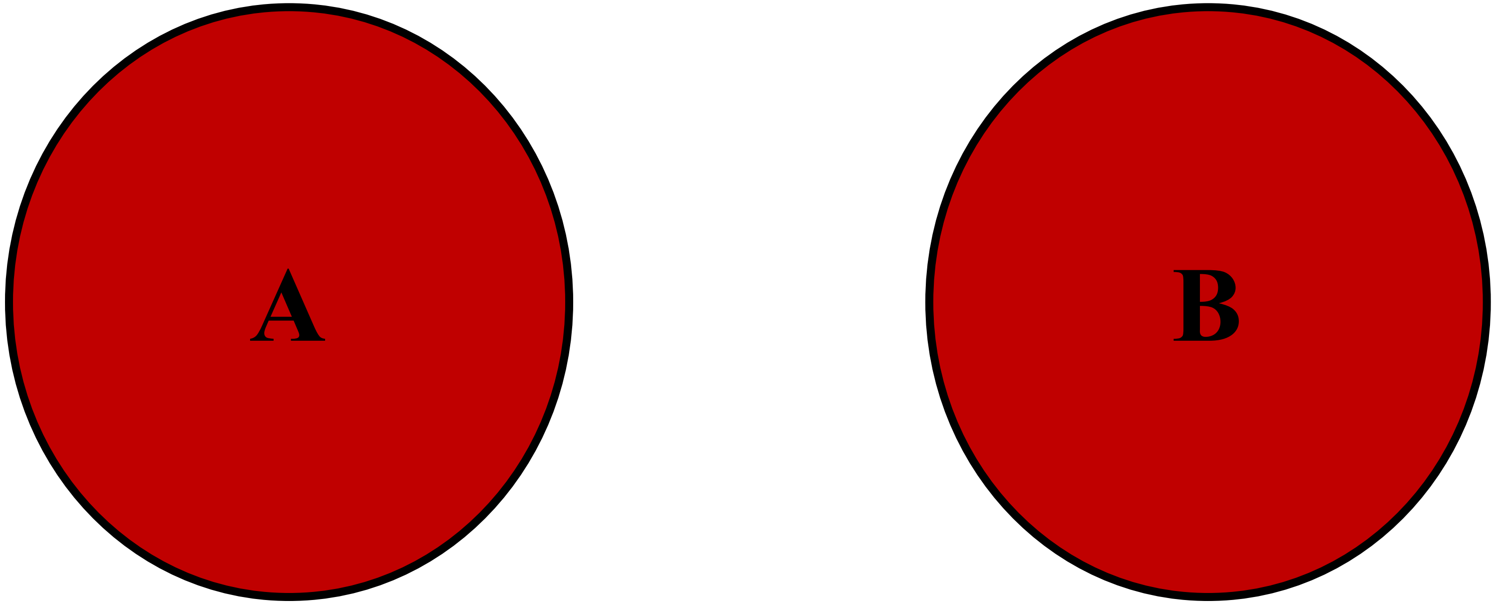
- Merge & joins
- Concat
- Clean

Group Exercise: European Trade

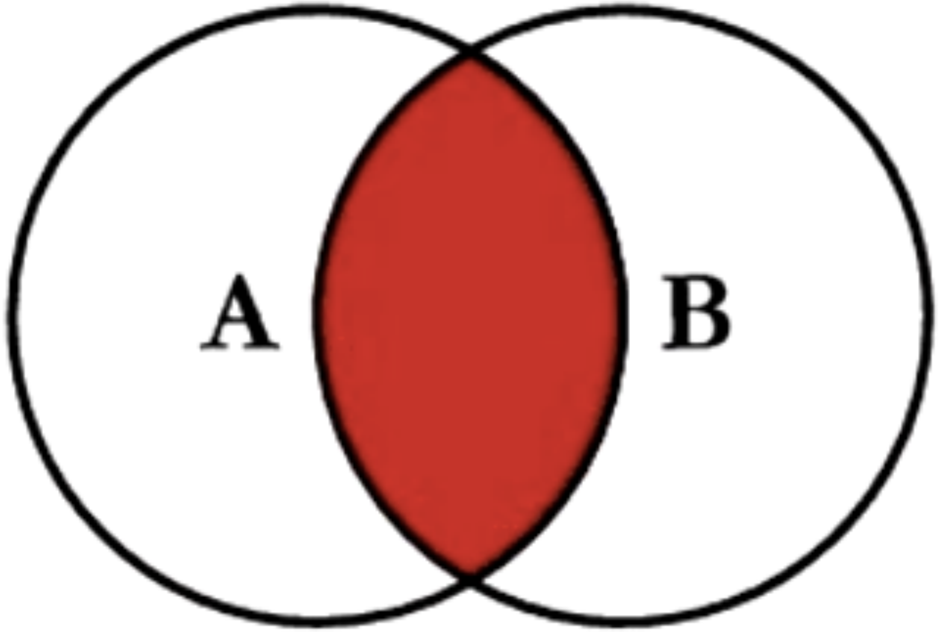
Big Data Processing with Spark

- Databricks CE

Joining Datasets

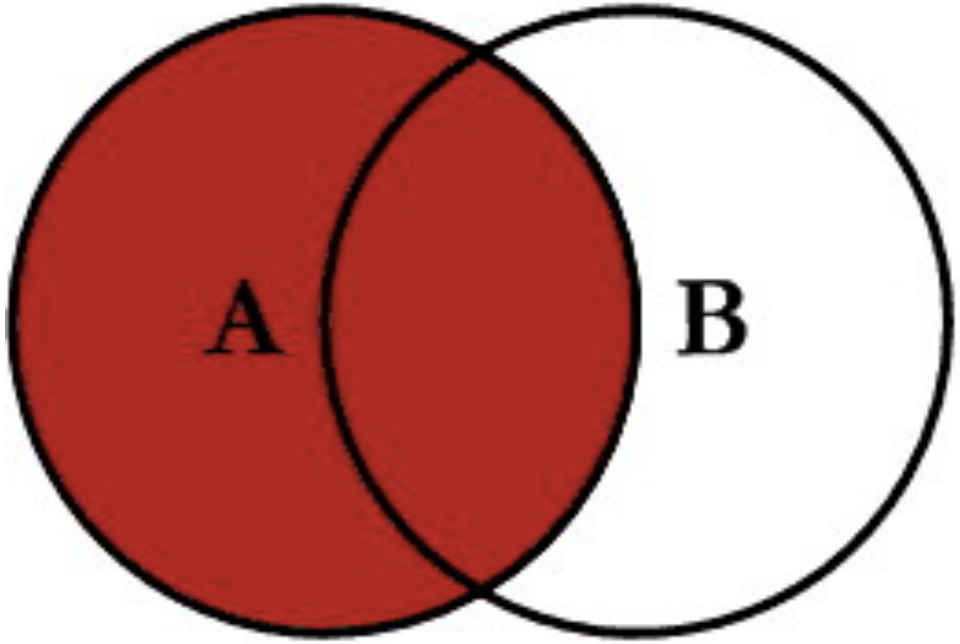


Joining Datasets : Inner Join



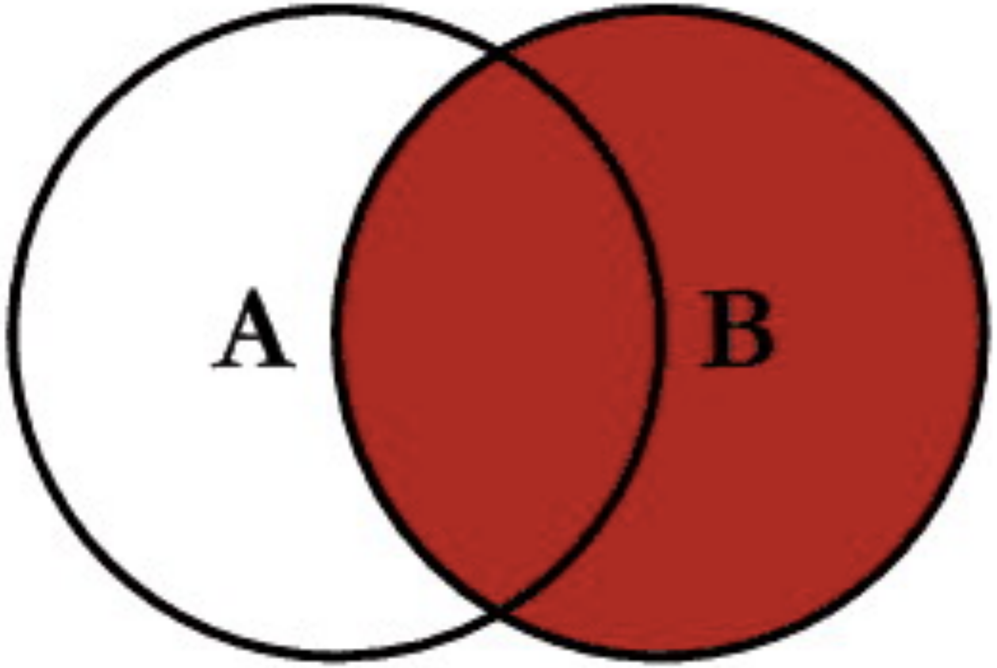

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

Joining Datasets : Left Outer Join



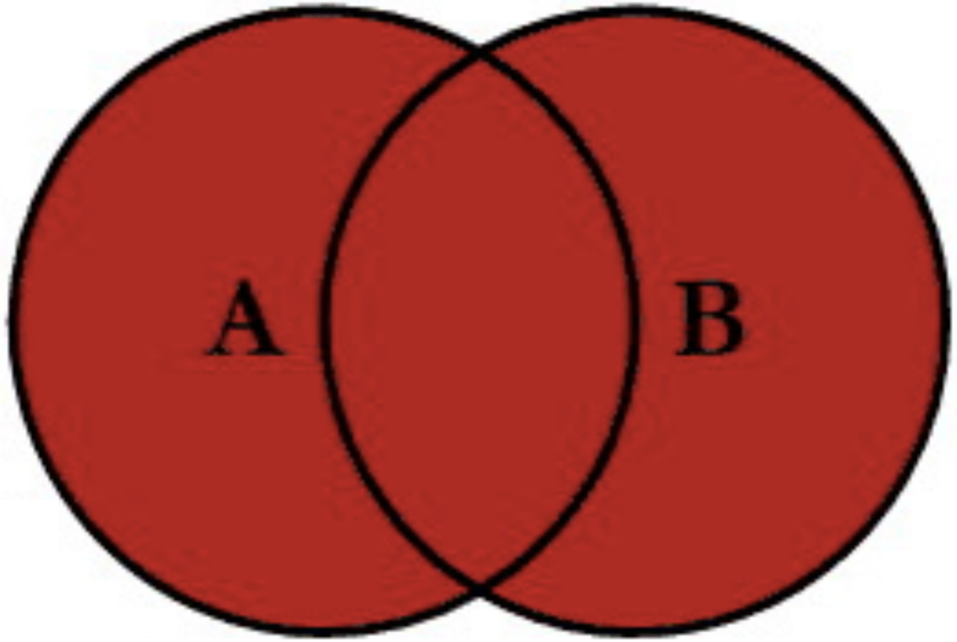

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

Joining Datasets : Right Outer Join




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

Joining Datasets : Full Outer Join




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

Merging (inner join)

data1	key
0	b
1	b
2	a
3	c
4	a
5	a
6	b

merge

data2	key
0	a
1	b
2	d

→

data1	key	data2
0	b	1
1	b	1
6	b	1
2	a	0
4	a	0
5	a	0

Merging (inner join)

data1	key
0	b
1	b
2	a
3	c
4	a
5	a
6	b

inner join

data2	key
0	a
1	b
2	d

→

data1	key	data2
0	b	1
1	b	1
6	b	1
2	a	0
4	a	0
5	a	0

Merging (left outer join)

data1	key
0	b
1	b
2	a
3	c
4	a
5	a
6	b

outer join

data2	key
0	a
1	b
2	d

→

data1	key	data2
0	b	1.0
1	b	1.0
2	a	0.0
3	c	NaN
4	a	0.0
5	a	0.0
6	b	1.0

Merging (right outer join)

data1	key
0	b
1	b
2	a
3	c
4	a
5	a
6	b

outer join

data2	key
0	a
1	b
2	d

→

data1	key	data2
0.0	b	1
1.0	b	1
6.0	b	1
2.0	a	0
4.0	a	0
5.0	a	0
NaN	d	2

Merging (full outer join)

data1	key
0	b
1	b
2	a
3	c
4	a
5	a
6	b

outer join

data2	key
0	a
1	b
2	d

→

data1	key	data2
0.0	b	1.0
1.0	b	1.0
6.0	b	1.0
2.0	a	0.0
4.0	a	0.0
5.0	a	0.0
3.0	c	NaN
NaN	d	2.0

Open notebook: “lecture06.data.shaping”

Agenda

Becoming a Data Ninja

- Merge & joins
- Concat
- Clean

Group Exercise: European Trade

Big Data Processing with Spark

- Databricks CE

Group Exercise: European Trade

European Union
circa 2016

Extra-EU Trade Data for 2010, 2012, 2014

http://ec.europa.eu/eurostat/web/products-datasets/-/ext_lt_invcur

“Extra-EU trade” statistics cover the trading of goods between Member States and a non-member countries.

SITC : Standard International Trade Classification

SITC0-4A

Extra-EU Trade Data for 2010, 2012, 2014

```
partner,currency,stk_flow,sitc06,geo\time      2014    2012    2010
EXT_EU,EUR,EXP,SITC0-4A,AT      61.9    65.6    67
EXT_EU,EUR,EXP,SITC0-4A,BE      53.8    85.8    92.4
EXT_EU,EUR,EXP,SITC0-4A,BG      57      46.2    54.1
EXT_EU,EUR,EXP,SITC0-4A,CY      79.1    60.7    61.4
EXT_EU,EUR,EXP,SITC0-4A,CZ      58.3    66.7    59.1
EXT_EU,EUR,EXP,SITC0-4A,DE      62.5    61.5    65.9
EXT_EU,EUR,EXP,SITC0-4A,DK      12.8    14      12.2
EXT_EU,EUR,EXP,SITC0-4A,EA      60.7    65.4    64.1
EXT_EU,EUR,EXP,SITC0-4A,EE      67.9    62.8    51.8
EXT_EU,EUR,EXP,SITC0-4A,EL      60.3    58.4    59
EXT_EU,EUR,EXP,SITC0-4A,ES      61.8    63.7    75.6
EXT_EU,EUR,EXP,SITC0-4A,EU      50.1    53.5    53.2
EXT_EU,EUR,EXP,SITC0-4A,FI      42.4    40.7    47.7
EXT_EU,EUR,EXP,SITC0-4A,FR      63.8    62.3    58.4
EXT_EU,EUR,EXP,SITC0-4A,HR      77.3    :       :
EXT_EU,EUR,EXP,SITC0-4A,HU      45.4    45.5    67.8
...
```

Read in by chunk of 100 rows

```
df = pd.DataFrame()

for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
       df = pd.concat([df, chunk])
```

	partner,currency,stk_flow,sitc06,geo\time	2014	2012	2010
0	EXT_EU,EUR,EXP,SITC0-4A,AT	61.9	65.6	67
1	EXT_EU,EUR,EXP,SITC0-4A,BE	53.8	85.8	92.4
2	EXT_EU,EUR,EXP,SITC0-4A,BG	57.0	46.2	54.1

Transforming column 1 : step 1 (splitting)

```
df = pd.DataFrame()

for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
        data_rows = [row for row in chunk.ix[:,0].str.split(',')]
        data_cols = chunk.columns[0].split(',')
        print(data_rows[:2], data_cols)
        break;
```

```
([['EXT_EU', 'EUR', 'EXP', 'SITC0-4A', 'AT'], ['EXT_EU', 'EUR', 'EXP',
'SITC0-4A', 'BE']], ['partner', 'currency', 'stk_flow', 'sitc06',
'geo\\time'])
```

Transforming column 1 : step 2 (fixing colname)

```
df = pd.DataFrame()

for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
        data_rows = [row for row in chunk.ix[:,0].str.split(',')]
        data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
        print(data_rows[:2], data_cols)
        break;
```

```
([['EXT_EU', 'EUR', 'EXP', 'SITC0-4A', 'AT'], ['EXT_EU', 'EUR', 'EXP',
'SITC0-4A', 'BE']], ['partner', 'currency', 'stk_flow', 'sitc06', 'geo'])
```

Transforming column 1 : step 3 (merge)

```
df = pd.DataFrame()
for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
       data_rows = [row for row in chunk.ix[:,0].str.split(',')]
       data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
       clean_df = pd.DataFrame(data_rows, columns=data_cols)

       # now we can concat by "column" which means axis=1
       new_df = pd.concat([clean_df, chunk], axis=1)
       print(new_df)
       break;
```

```
partner currency stk_flow sitc06 geo \
0 EXT_EU EUR EXP SITC0-4A AT
1 EXT_EU EUR EXP SITC0-4A BE
2 EXT_EU EUR EXP SITC0-4A BG
3 EXT_EU EUR EXP SITC0-4A CY
4 EXT_EU EUR EXP SITC0-4A CZ
```

Transforming column 1 : step 4 (clean)

```
df = pd.DataFrame()
for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
        data_rows = [row for row in chunk.ix[:,0].str.split(',')]
        data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
        clean_df = pd.DataFrame(data_rows, columns=data_cols)

        # now we can concat by "column" which means axis=1
        new_df = pd.concat([clean_df, chunk.drop(chunk.columns[0], axis=1)],
                                axis=1)
        print(new_df)
        break;
```

```
partner currency stk_flow sitc06 geo
0 EXT_EU EUR EXP SITC0-4A AT
1 EXT_EU EUR EXP SITC0-4A BE
2 EXT_EU EUR EXP SITC0-4A BG
3 EXT_EU EUR EXP SITC0-4A CY
4 EXT_EU EUR EXP SITC0-4A CZ
```

Transforming column 1 : step 5 (finalize)

```
df = pd.DataFrame()
for chunk in pd.read_csv('data/ext_lt_invcur.tsv', sep='\t', chunksize=100):
       data_rows = [row for row in chunk.ix[:,0].str.split(',')]
       data_cols = [col.split('\\')[0] for col in chunk.columns[0].split(',')]
       clean_df = pd.DataFrame(data_rows, columns=data_cols)

       # now we can concat by "column" which means axis=1
       new_df = pd.concat([clean_df, chunk.drop(chunk.columns[0], axis=1)],
                          axis=1)
       df = pd.concat([df, new_df])
```

	partner	currency	stk_flow	sitc06	geo	2014	2012	2010
0	EXT_EU	EUR	EXP	SITC0-4A	AT	61.9	65.6	67
1	EXT_EU	EUR	EXP	SITC0-4A	BE	53.8	85.8	92.4
2	EXT_EU	EUR	EXP	SITC0-4A	BG	57.0	46.2	54.1

Data Exploration

```
df.shape()
```

```
(1320, 8)
```

Data Exploration

```
df.describe(include='all')
```

	partner	currency	stk_flow	sitc06	geo	2014	2012	2010
count	1320	1320	1320	1320	1320	1320.000000	1320	1320
unique	2	5	2	4	33	NaN	518	471
top	EXT_EU	OTH	IMP	SITC5-8	UK	NaN	100	100
freq	1200	264	660	330	40	NaN	238	248
mean	NaN	NaN	NaN	NaN	NaN	39.998712	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	39.025858	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	2.275000	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	28.650000	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	75.800000	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	100.000000	NaN	NaN

Agenda

Becoming a Data Ninja

- Merge & joins
- Concat
- Clean

Group Exercise: European Trade

Big Data Processing with Spark

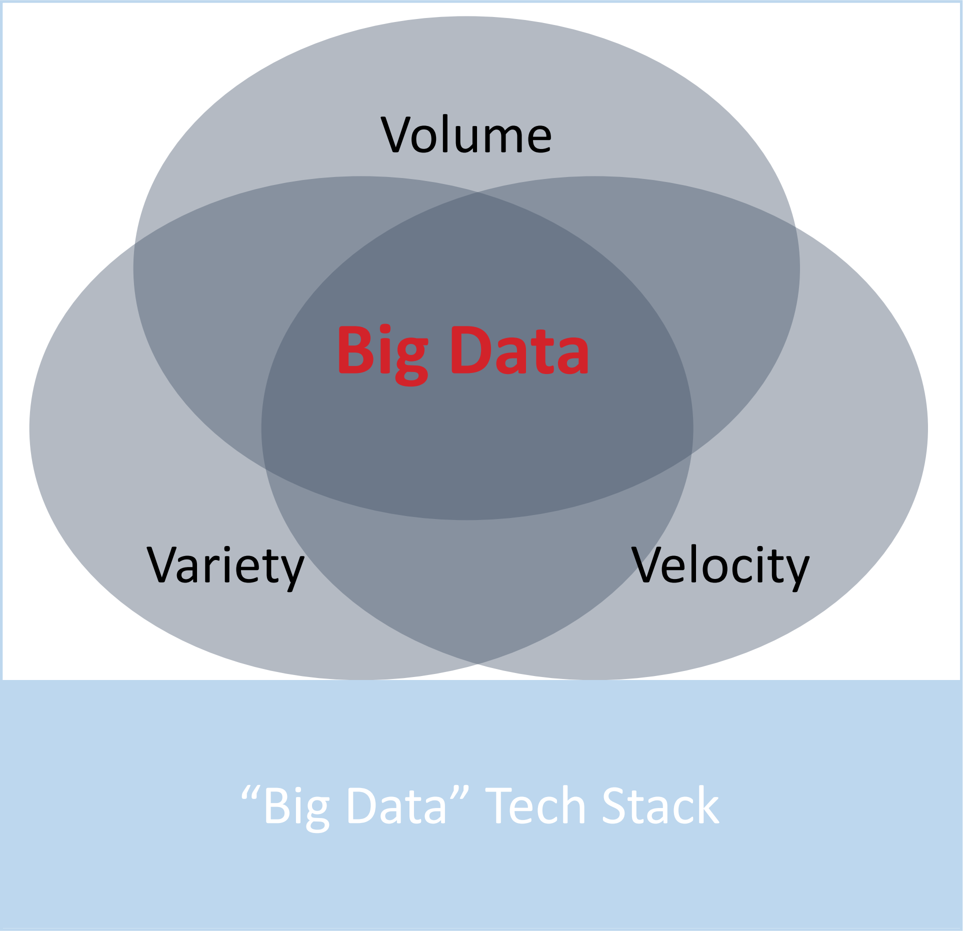
- Databricks CE

Big Data Processing with

What is Apache Spark?

A **distributed computing** framework
for **scalable**, **efficient** analysis of big data.

What is Big Data?



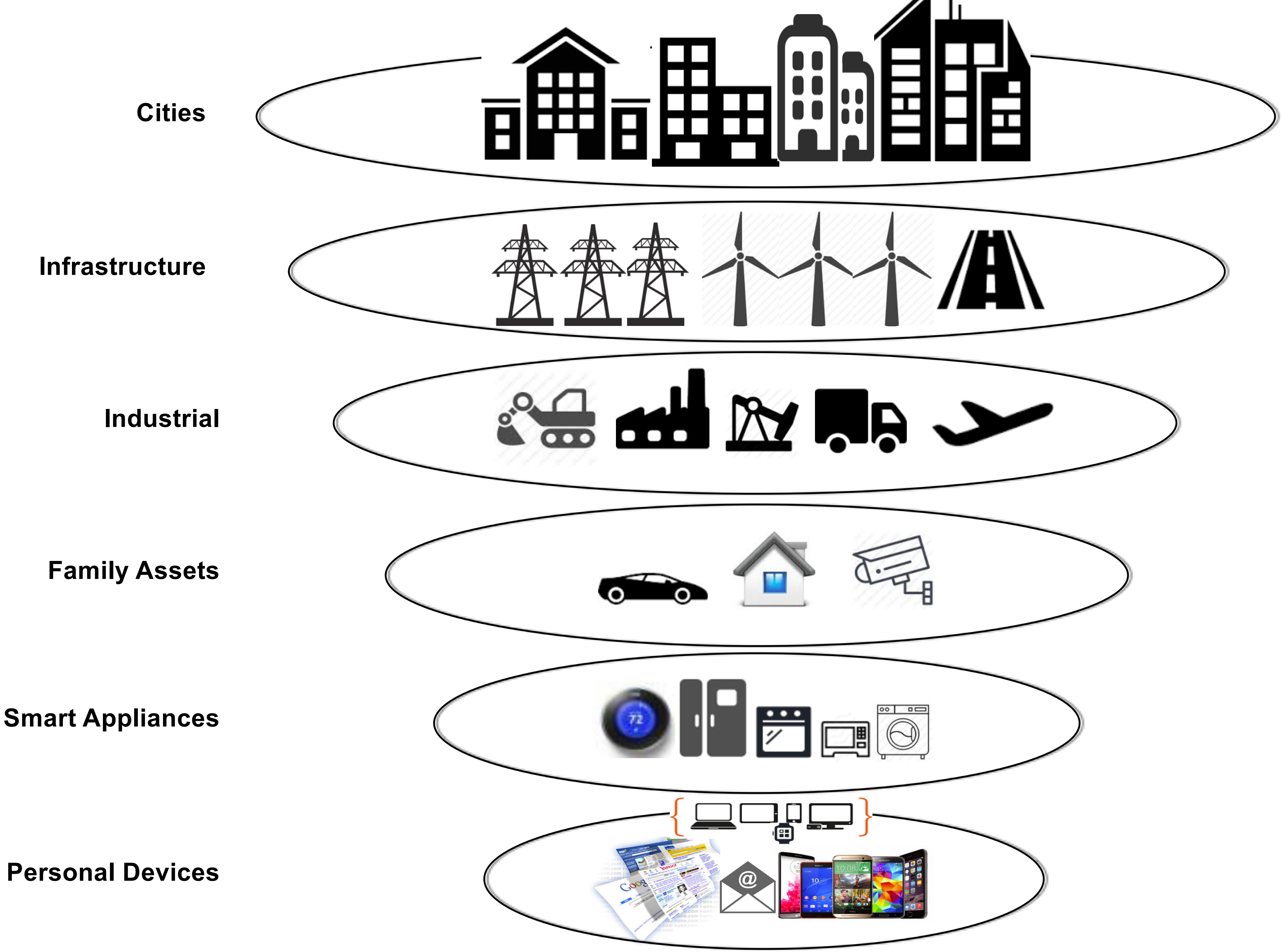

msn
YAHOO!


msn
YAHOO!


SONY
10:08


msn
YAHOO!


SONY
10:08
HTC


Cities
Infrastructure
Industrial
Family Assets
Smart Appliances
Personal Devices

The Big Data Problem

- 1 machine cannot store nor process all the data
- Idea: distribute data in ”clusters of machines” for “distributed computing”

Distributed Computing : Word Count

"I am Sam; I am Sam."

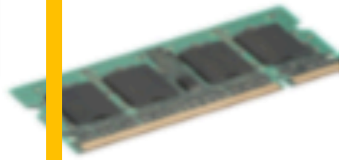
Distributed Computing : Word Count

Word	Index	Count
I	0	1
am	2	1
Sam	5	1
I	9	1
am	11	1
Sam	14	1

DataFrame

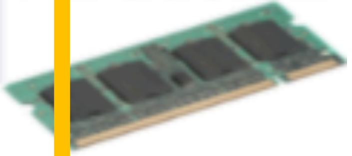
I	0	1
am	2	1

Partition 1



Sam	5	1
I	9	1

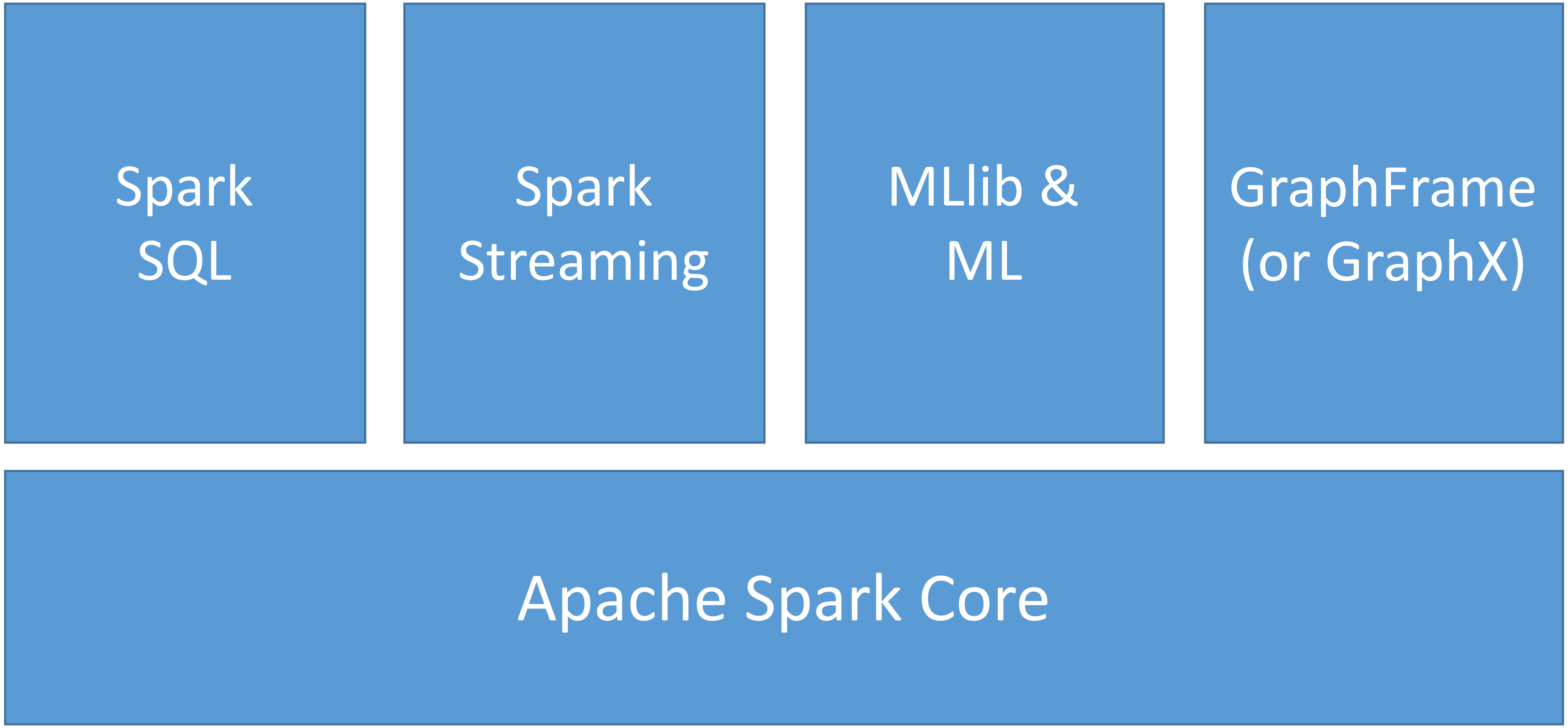
Partition 2



am	11	1
Sam	14	1

Partition 3

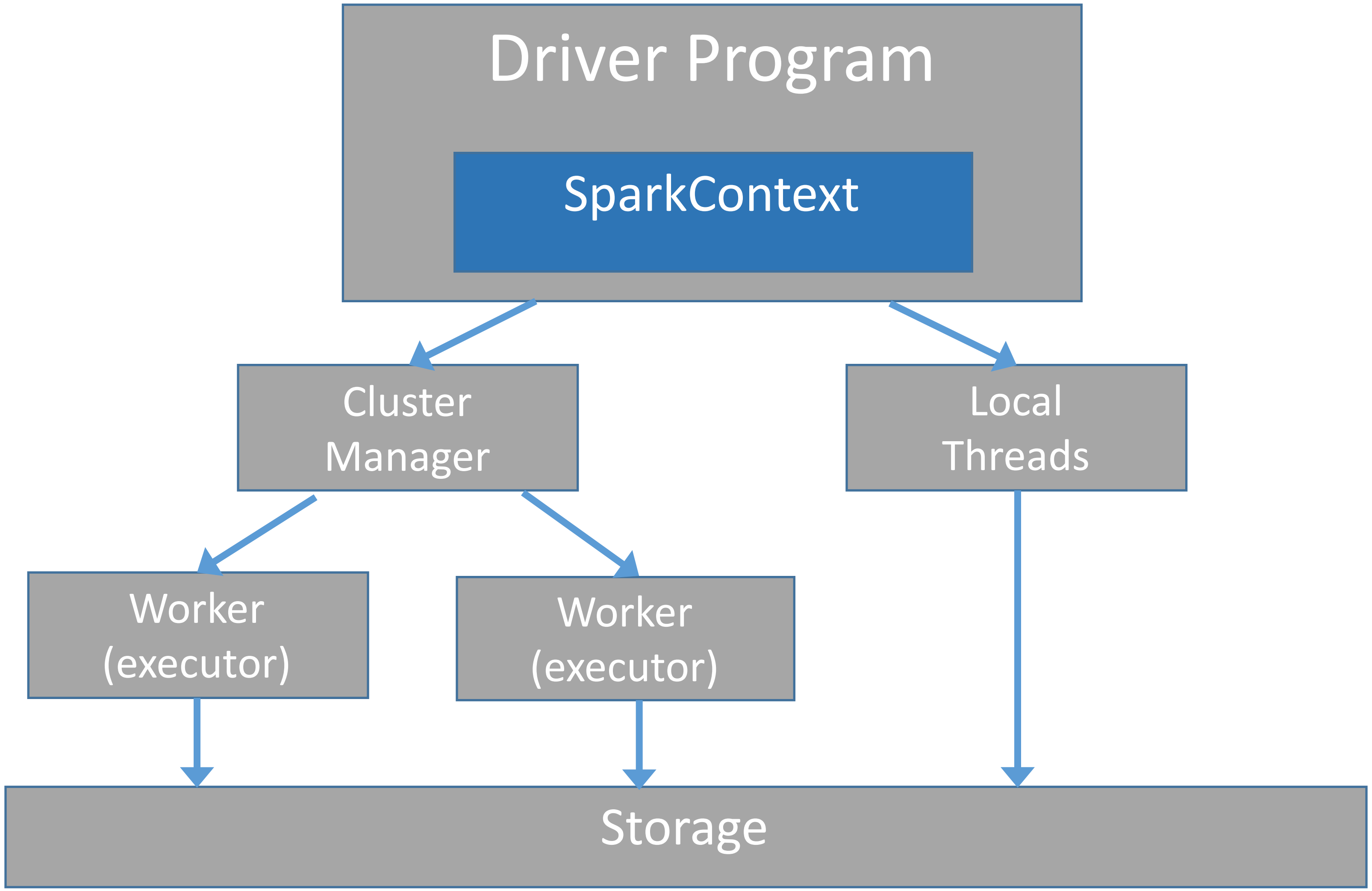
Apache Spark Components



Python Spark

- We use the Python programming interface to Spark (pySpark)

Spark Driver & Workers



SparkContext

- When a Spark program starts, it creates a `SparkContext` object
  - SparkContext tells Spark how and where to access a cluster
  - pySpark shell, Databricks CE automatically create SparkContext
  - iPython and user created programs must create a new SparkContext
- The program next creates a sqlContext object
  - Use `sqlContext` to create `DataFrames`

Spark DataFrame

- Not the same as a pandas DataFrame!
- Spark DataFrame: the primary data abstraction in Spark
  - Immutable once constructed
  - Track lineage information to efficiently recompute lost data
  - Enable operations on collection of elements in parallel
- Create a DataFrame by:
  - by *parallelizing* existing Python collections (lists)
  - by *transforming* an existing Spark or pandas DFs
  - from *files* in HDFS or any other storage system

Spark DataFrame

- Each row of a DataFrame is a Row object

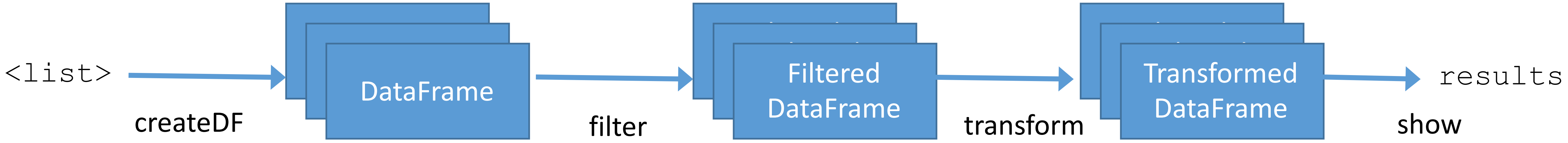
- The fields in a Row can be accessed like attributes

```
>>> row = Row(name="Alice", age=11)
>>> row
Row(age=11, name='Alice')
>>> row['name'], row['age'] ('Alice', 11)
>>> row.name, row.age ('Alice', 11)
```

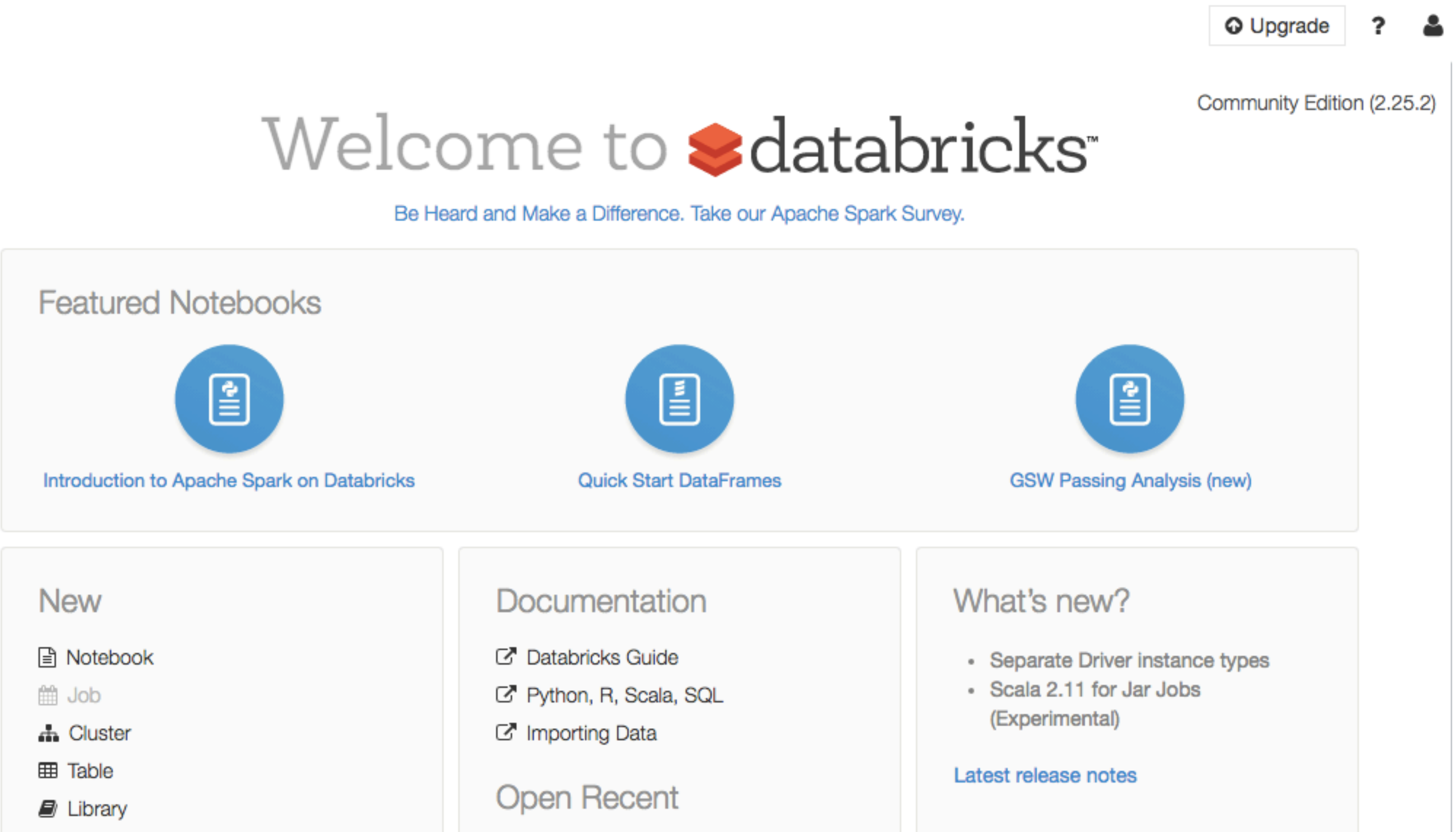
DataFrame: 2 types of ops

- Transformations
  - Transformations are lazy (*not computed immediately*)
  - Transformed DF is executed when action runs on it
  - Persist (cache) DFs in memory or disk

- Actions : collect, show, reduce, …

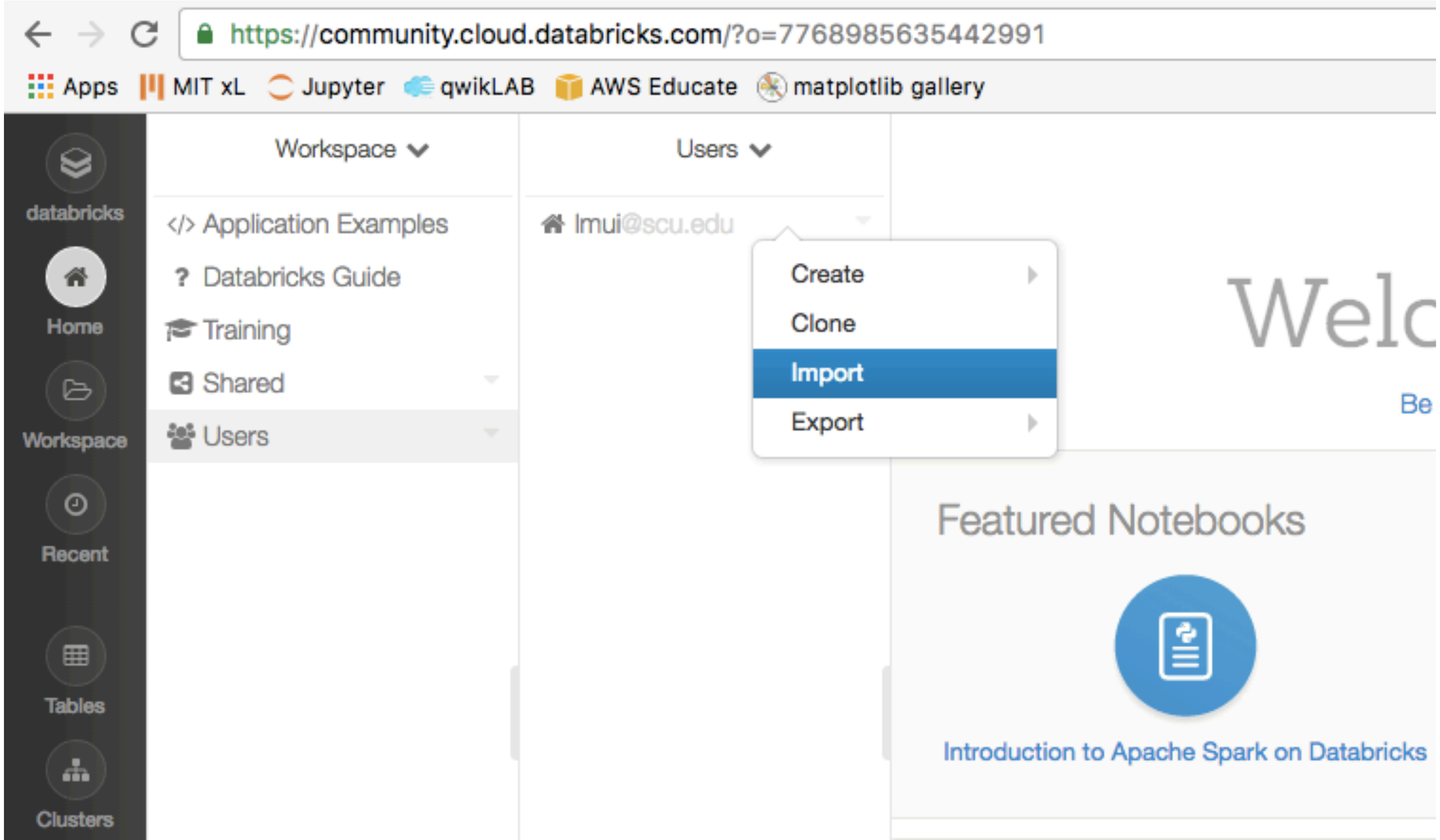


Spark / Databricks Community Edition



https://databricks.com/ce

Upload “lecture06.intro.spark.dbc”



Next week

- Complete work of Shakespeare
- NASA Weblog analysis