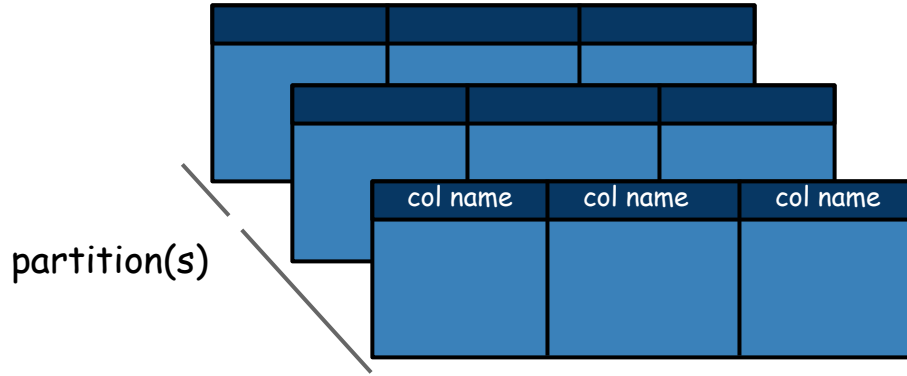


pyspark-pictures data frames

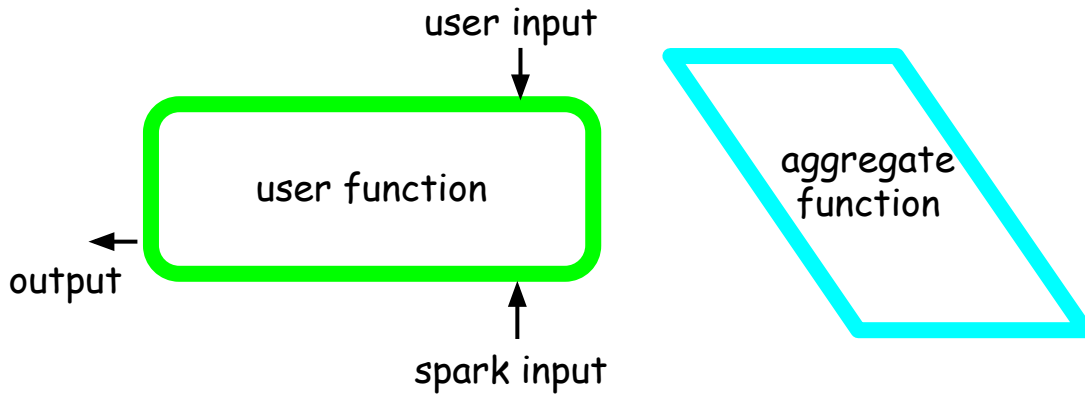
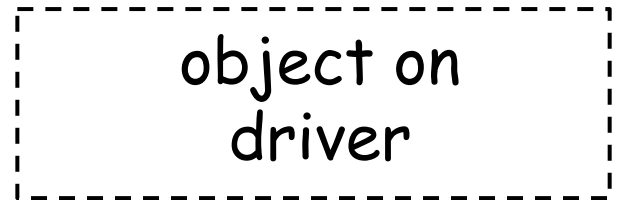
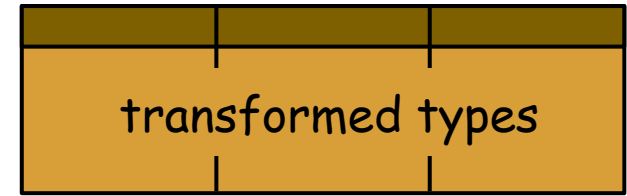
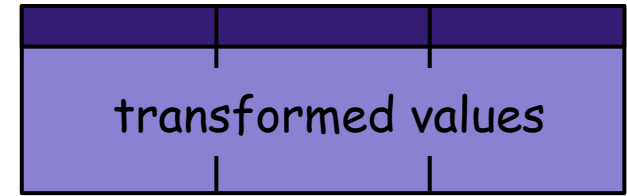
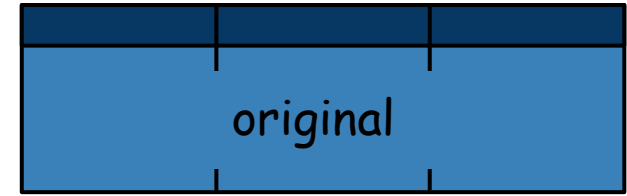
Learn the pyspark API through pictures and simple examples

<https://github.com/jkthompson/pyspark-pictures>

data frame



data frame Row



•

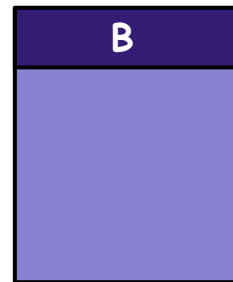
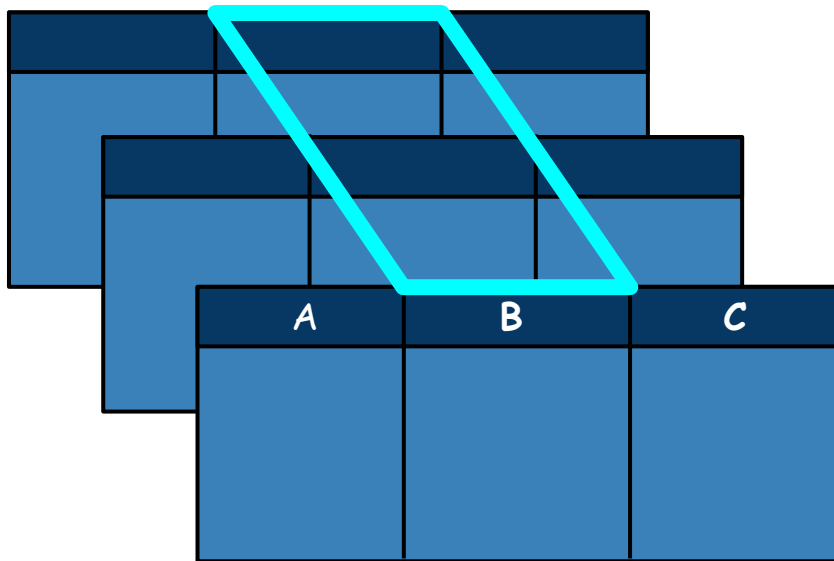
df.B

A	B	C

B

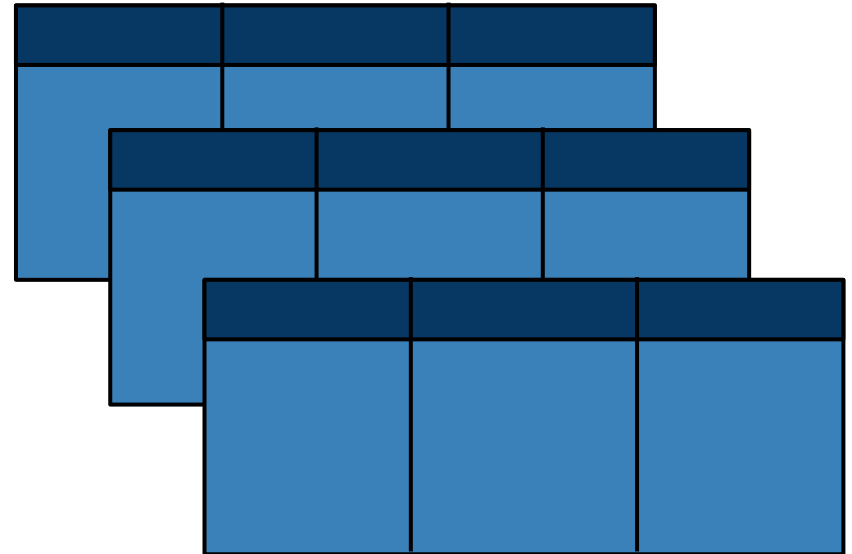
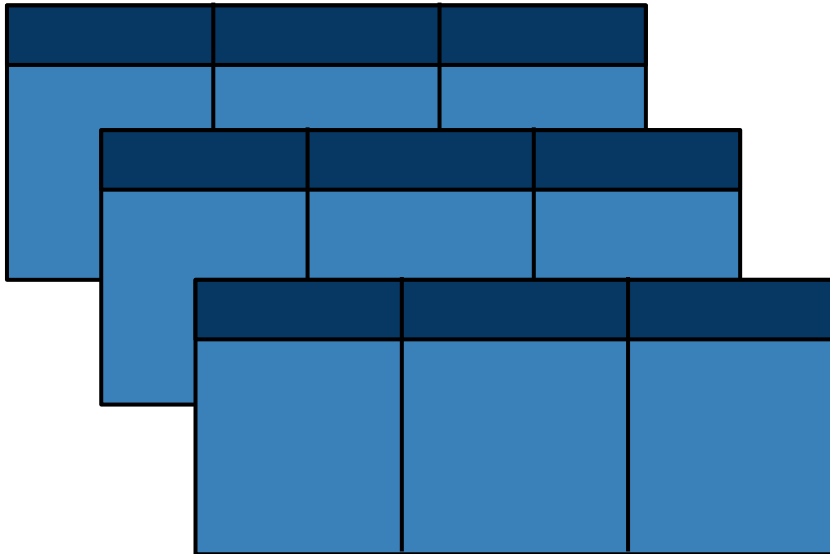
agg

`exprs = {"B" : "aggF"}`



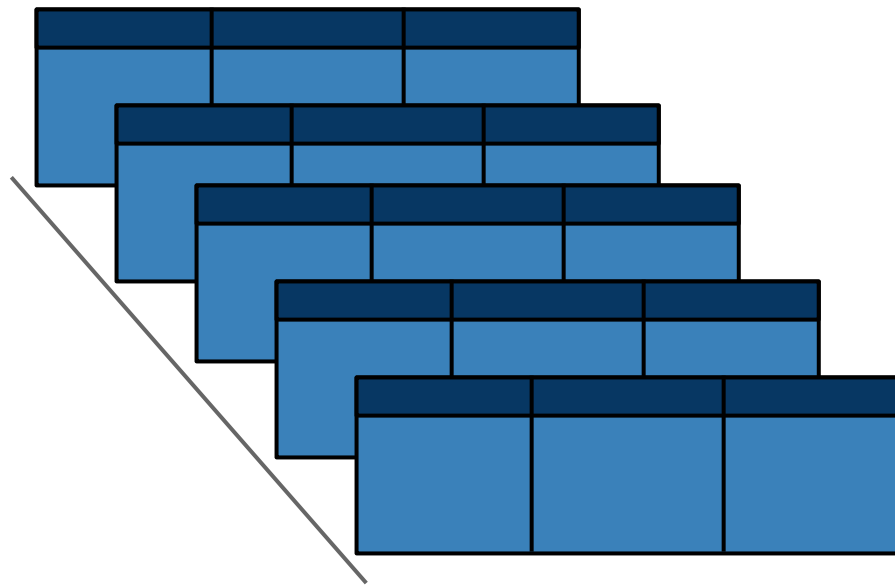
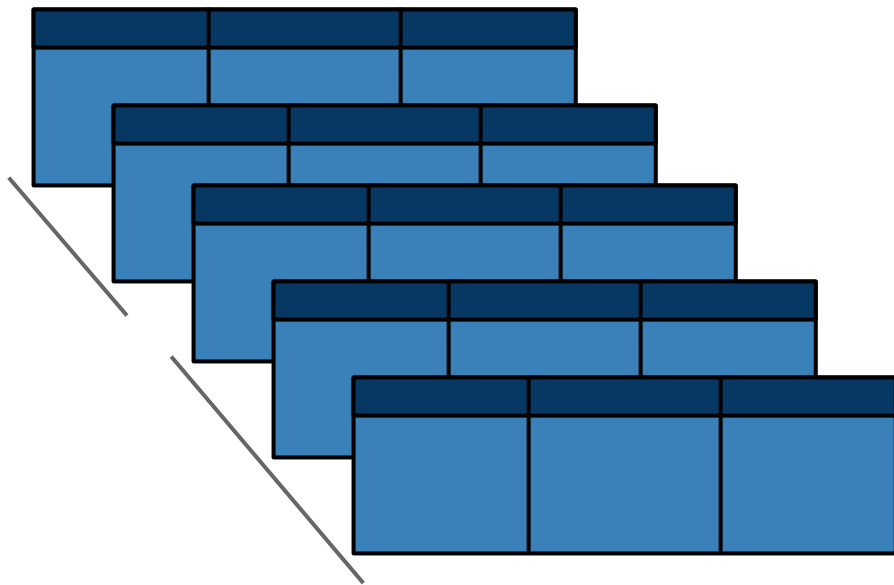
alias

alias = "foo"

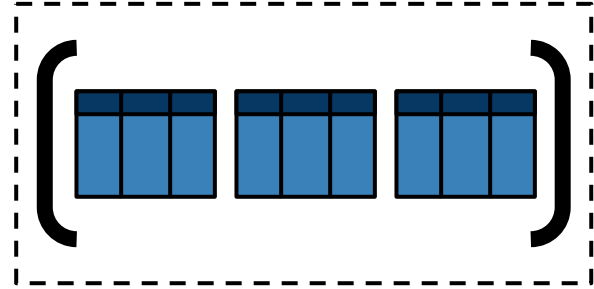
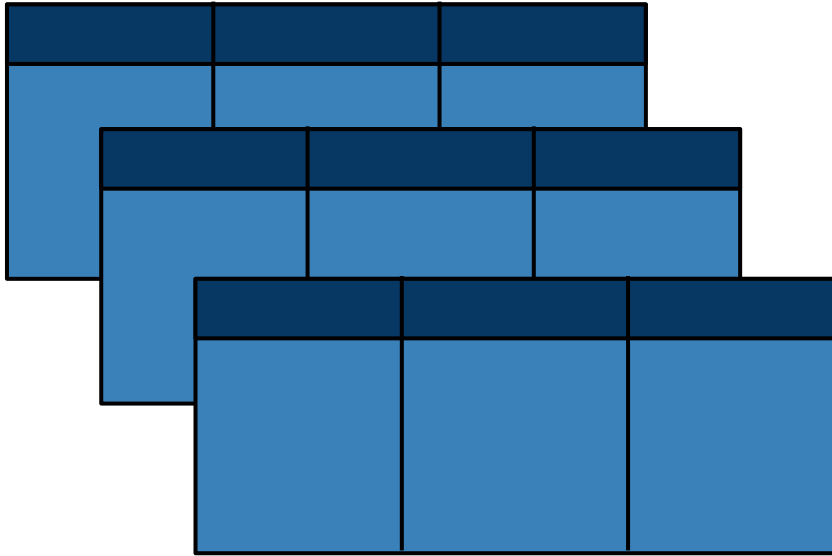


coalesce

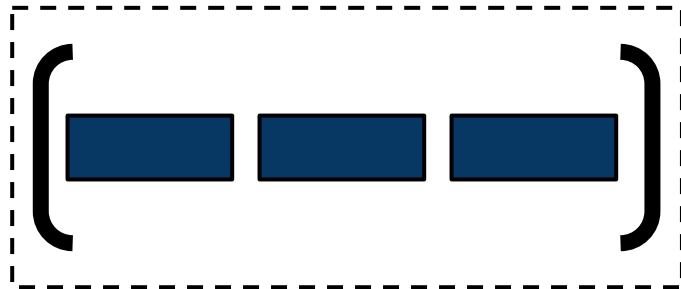
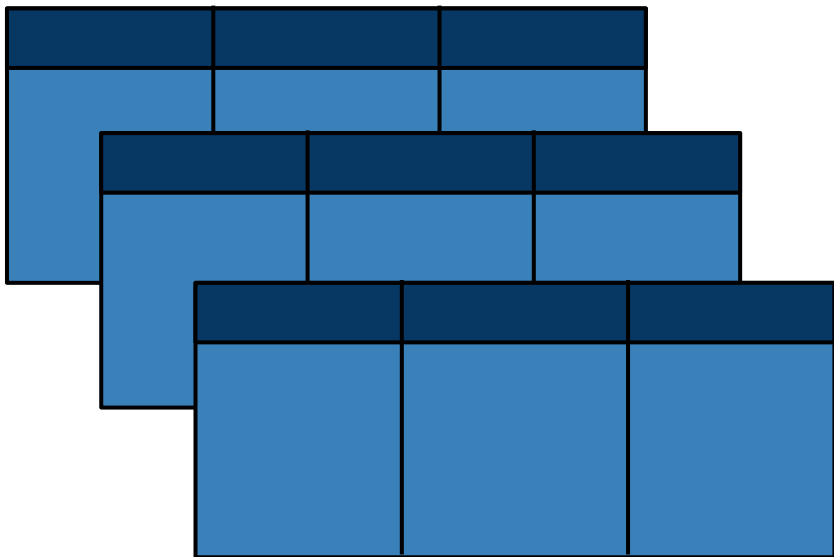
numPartitions = 1



collect



columns



corr

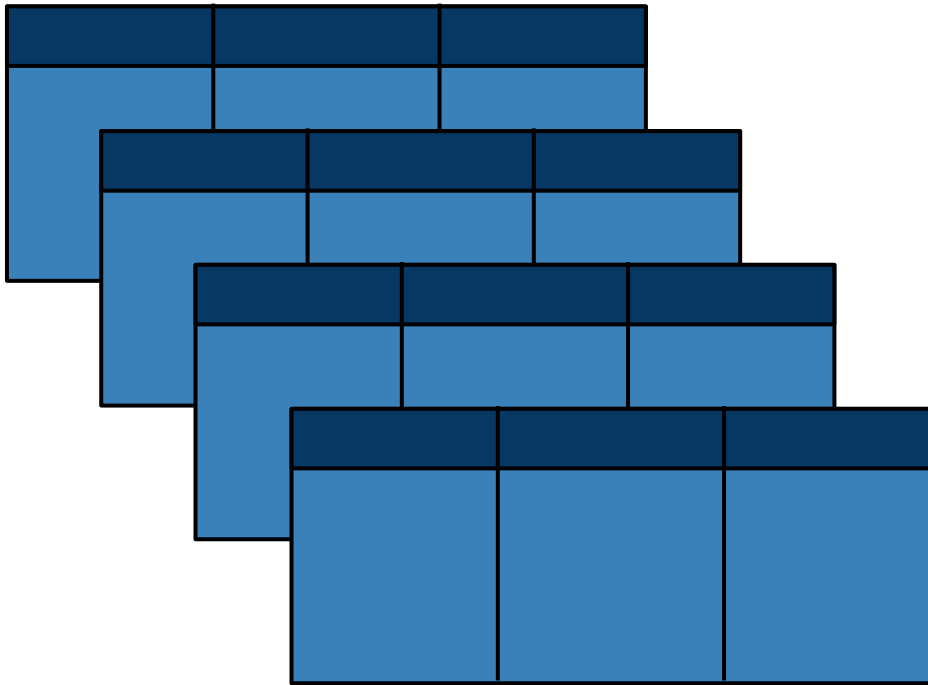
col1 = A col2 = C

	A	B	C

Pearson's r

$$r = \frac{\sum_i (A_i - \bar{A})(C_i - \bar{C})}{\sqrt{\sum_i (A_i - \bar{A})^2} \sqrt{\sum_i (C_i - \bar{C})^2}}$$

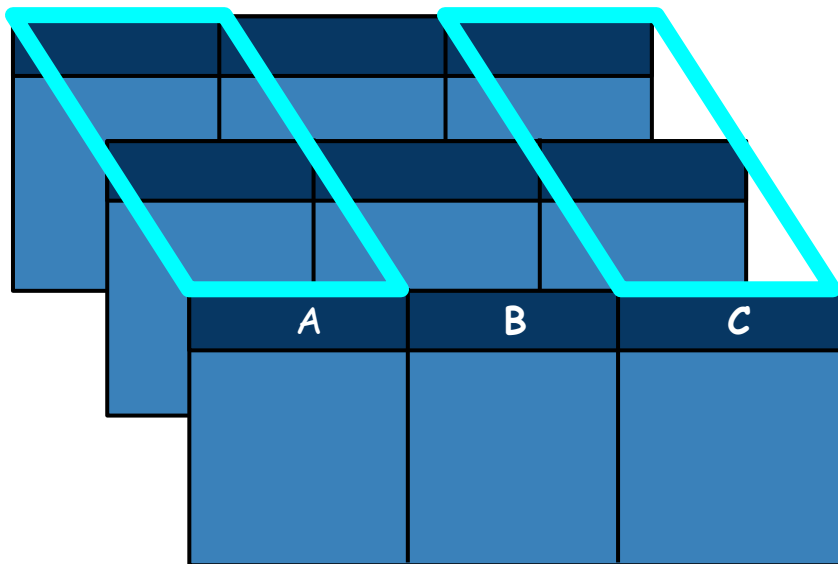
count



4

COV

col1 = A col2 = C



Sample Covariance

$$\frac{1}{N-1} \sum_i (A_i - \bar{A})(C_i - \bar{C})$$

crosstab

col1 = A col2 = C

	A	B	C
foo	baz	bar	baz
bar			
baz			

	A_C	foo	baz
foo	1	0	
bar	0	1	
baz	0		1

cube

col1 = A col2 = C

	A	B	C
1	2	20	200
10			
100			

null	100			
null	null			
1	null			
null	200			
2	null			
2	200			
A	C	agg(A)	agg(B)	agg(C)
1	100			

describe

```
cols = ['A','B']
```

The diagram illustrates a 3D grid of blue rectangular blocks. Three parallel planes are highlighted with cyan outlines and labeled A, B, and C from top to bottom. The grid is composed of blue rectangular blocks arranged in a 3x3x3 structure.

[illegible]

distinct

foo	foo	foo
bar	foo	baz
A	B	C
bar	foo	baz

foo	foo	foo
A	B	C
bar	foo	baz

drop

col = 'A'

foo	foo	foo
bar	foo	baz
A	B	C
bar	foo	baz

foo	foo
foo	baz
B	C
foo	baz

dropDuplicates

subset = ['B']

foo	foo	foo
bar	foo	baz
A	B	C
bar	foo	baz

foo	foo	foo
A	B	C
bar	foo	baz

drop_duplicates

subset = ['B']

foo	foo	foo
bar	foo	baz
A	B	C
bar	foo	baz

foo	foo	foo
A	B	C
bar	foo	baz

dropna

how = 'any' subset = ['A', 'B']

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

A	B	C
bar	foo	null

dtypes

foo	3	0.2

bar	2	1.0

A	B	C
bar	1	2.2

[('A','string'), ('B','int'), ('C','float')]

explain

extended = True

foo	3	0.2

bar	2	1.0

A	B	C
bar	1	2.2

== Parsed Logical Plan ==

...

== Analyzed Logical Plan ==

...

== Optimized Logical Plan ==

...

== Physical Plan ==

...

== RDD ==

fillna

value = 'unknown' subset = ['A', 'B']

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

foo	unknown	foo

unknown	foo	baz

A	B	C
bar	foo	null

filter

condition = "A = foo"

foo	foo	foo

bar	foo	baz

A	B	C
bar	foo	baz

A	B	C
foo	foo	foo

first

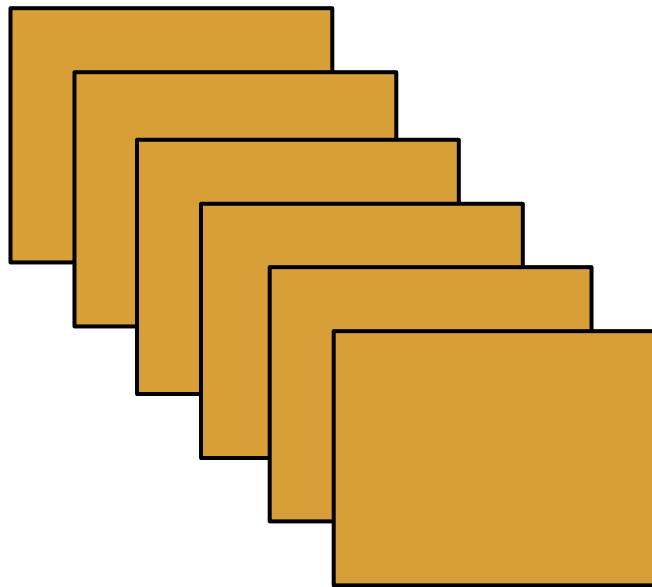
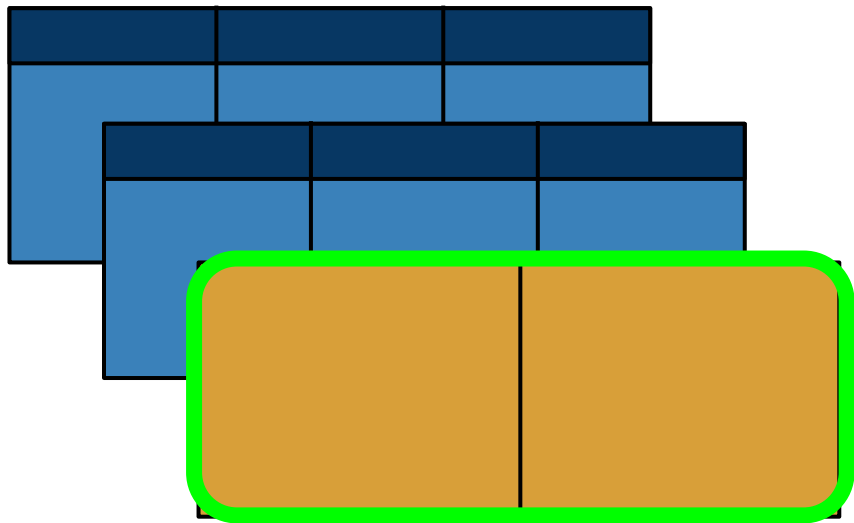
foo	foo	foo

bar	foo	baz

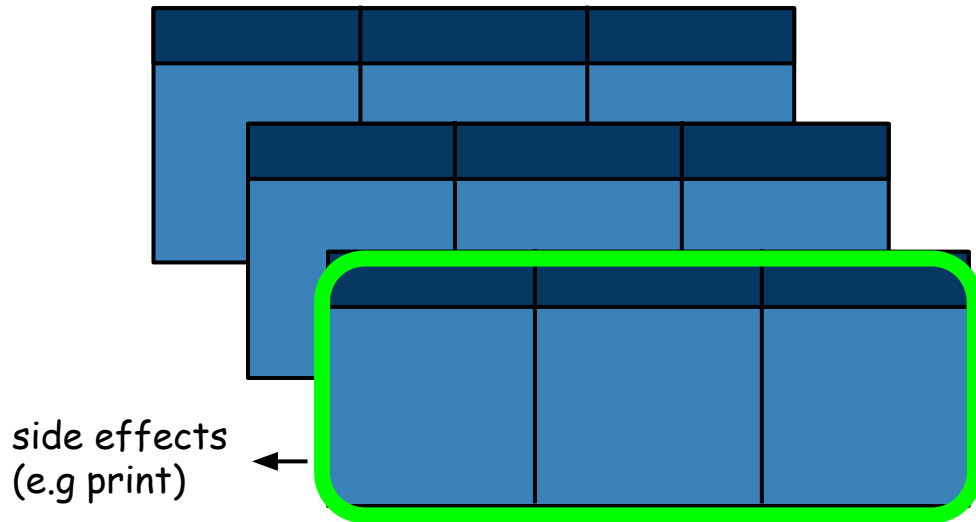
A	B	C
bar	foo	baz

Row(A='bar', B='foo', C='baz')

flatMap

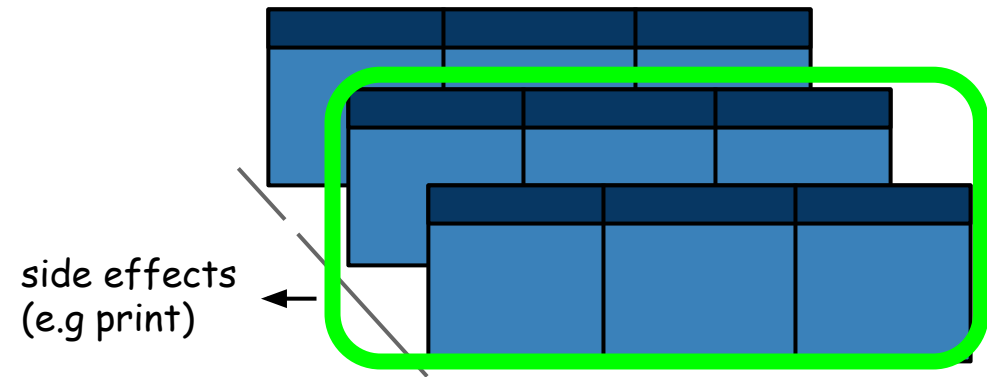


foreach



*no return value,
original DataFrame
unchanged

foreachPartition



*no return value,
original DataFrame
unchanged

freqItems

cols = ['A','C'] support = 0.5

foo	foo	foo
bar	foo	baz
A	B	C
bar	foo	baz

A_freqItems	C_freqItems
[bar,foo]	[baz,foo]

groupBy

['A', 'C']

	['A', 'C']		
	A	B	C
1	2	20	200
10			
100			

1	100				
A	C	agg(A)	agg(B)	agg(C)	
2	200				

groupby

['A', 'C']

	A	B	C
1	2	20	200
10			
100			

1	100			
A	C	agg(A)	agg(B)	agg(C)
2	200			

head

n = 2

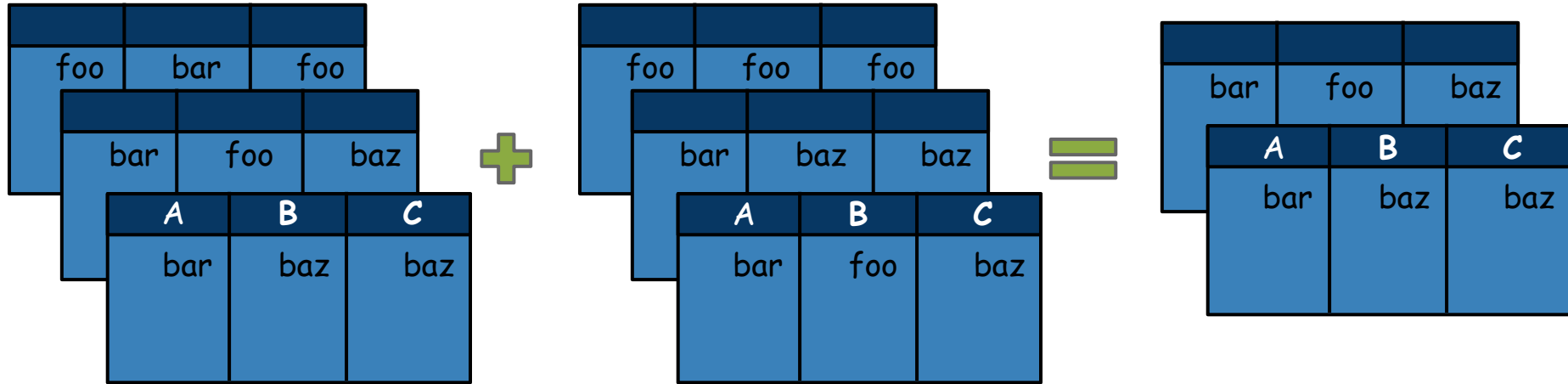
foo	foo	foo

bar	bar	baz

A	B	C
bar	foo	baz

[Row(A='bar', B='foo', C='baz'),
Row(A='bar', B='bar', C='baz')]

intersect



isLocal

The diagram illustrates three overlapping tables, each with a different set of column headers and data. The tables are arranged in a staggered, overlapping fashion.

- Table 1 (Top):** Has three columns, all with the header "foo".
- Table 2 (Middle):** Has three columns with headers "bar", "bar", and "baz".
- Table 3 (Bottom):** Has three columns with headers "A", "B", and "C".

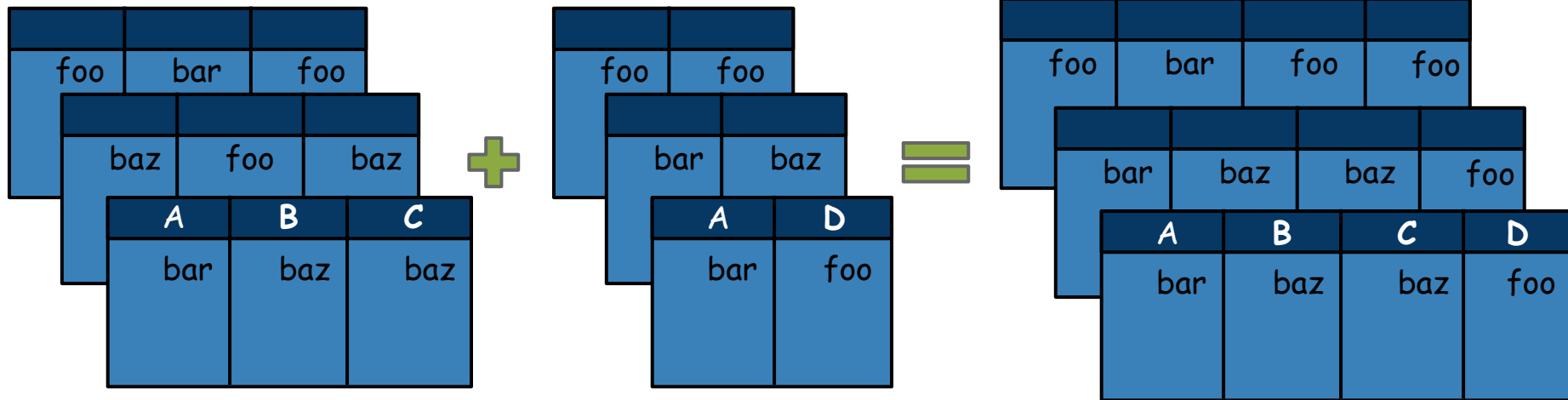
The data rows for each table are as follows:

- Table 1 Data:** All three data cells contain the value "foo".
- Table 2 Data:** The first two data cells contain "bar", and the third data cell contains "baz".
- Table 3 Data:** The data cells contain "bar", "foo", and "baz" respectively.

True

join

joinExprs = 'A', joinType = 'inner'



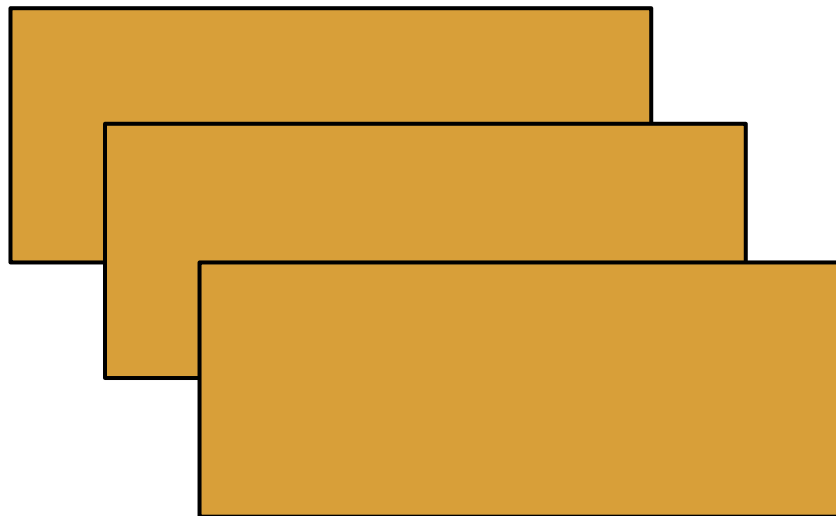
limit

num = 2

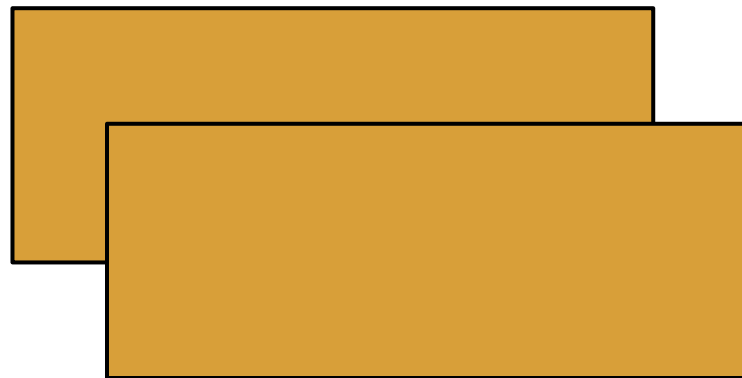
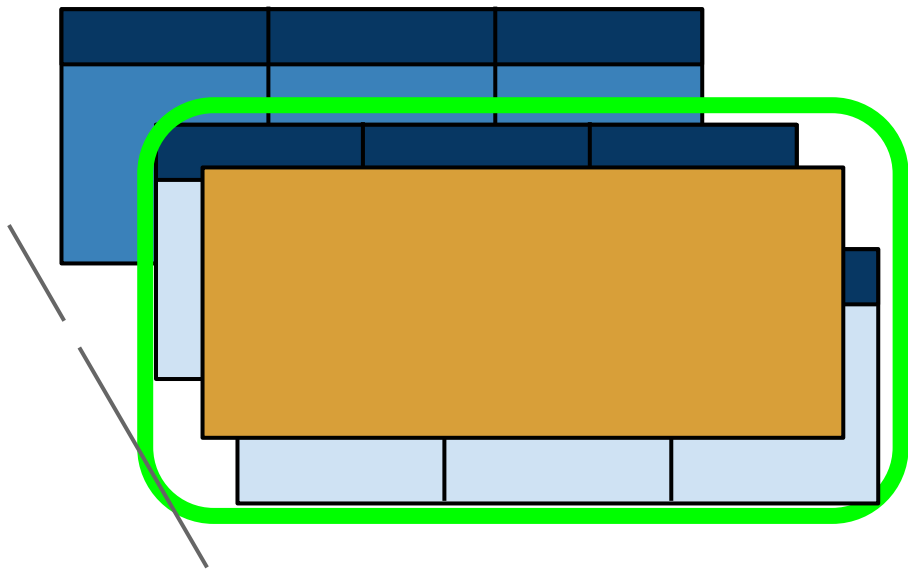
foo	foo	foo
bar	foo	baz
A	B	C
bar	foo	baz

bar	foo	baz
A	B	C
bar	foo	baz

map



mapPartitions



na

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

orderBy

```
cols = ['A', 'C'], ascending = [True, False]
```

The diagram illustrates three overlapping tables. The top table has three columns: 'baz', 'baz', and '3'. The middle table has three columns: 'bar', 'foo', and '1'. The bottom table has three columns: 'A', 'B', and 'C', with rows 'baz', 'foo', and '2'.

The diagram illustrates three overlapping tables, each with a different set of column headers and data. The tables are arranged in a staggered, overlapping fashion.

- Table 1 (Top):** Has three columns with headers (empty dark blue cells) and data (light blue cells). The data values are "baz", "foo", and "2".
- Table 2 (Middle):** Has three columns with headers (empty dark blue cells) and data (light blue cells). The data values are "baz", "baz", and "3".
- Table 3 (Bottom):** Has three columns with headers "A", "B", and "C" (dark blue cells) and data (light blue cells). The data values are "bar", "foo", and "1".

The tables overlap such that the right side of Table 1 is above the left side of Table 2, and the right side of Table 2 is above the left side of Table 3.

persist

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

printSchema

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

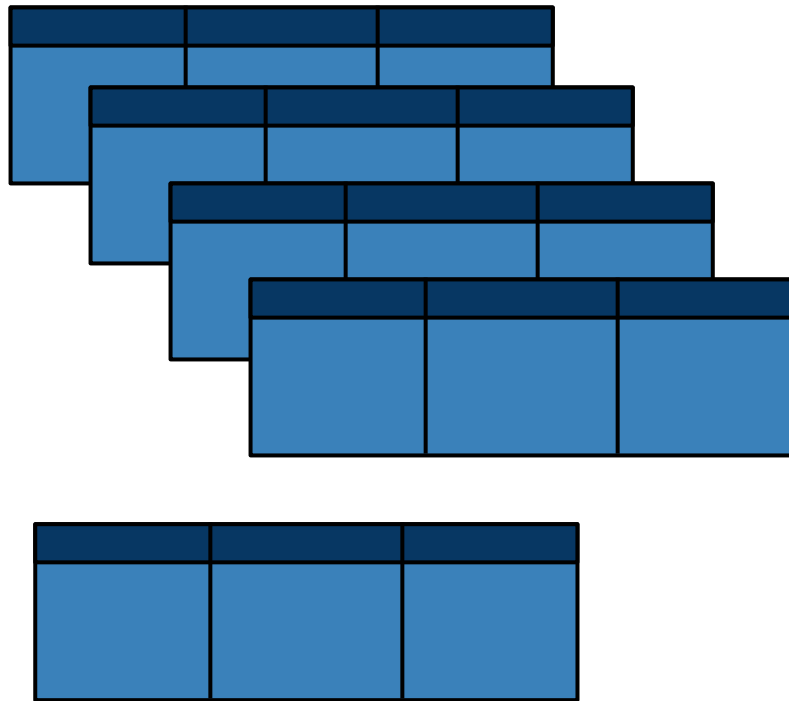
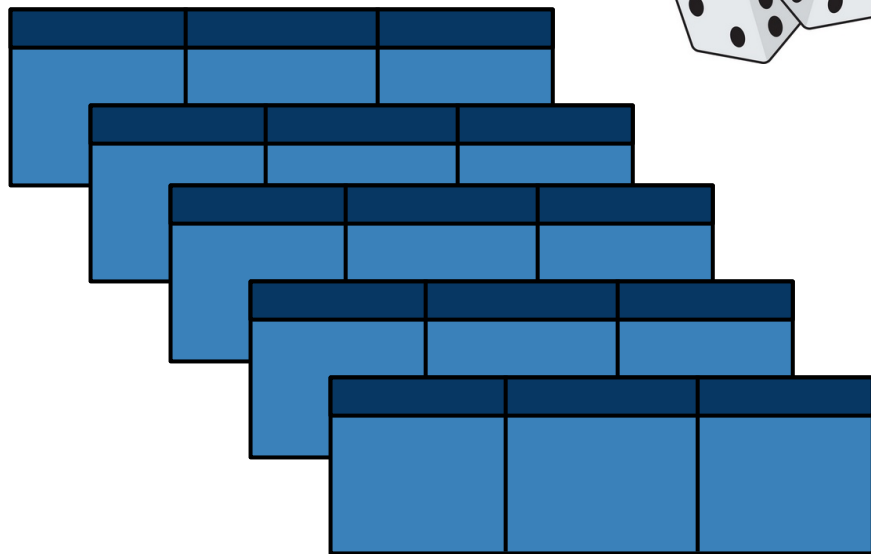
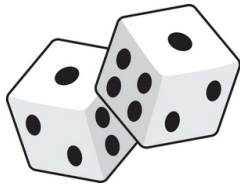
stdout

root

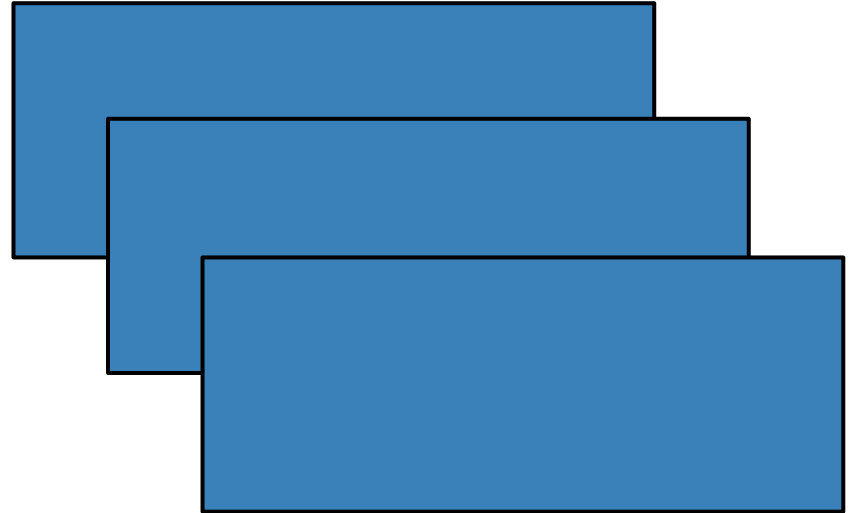
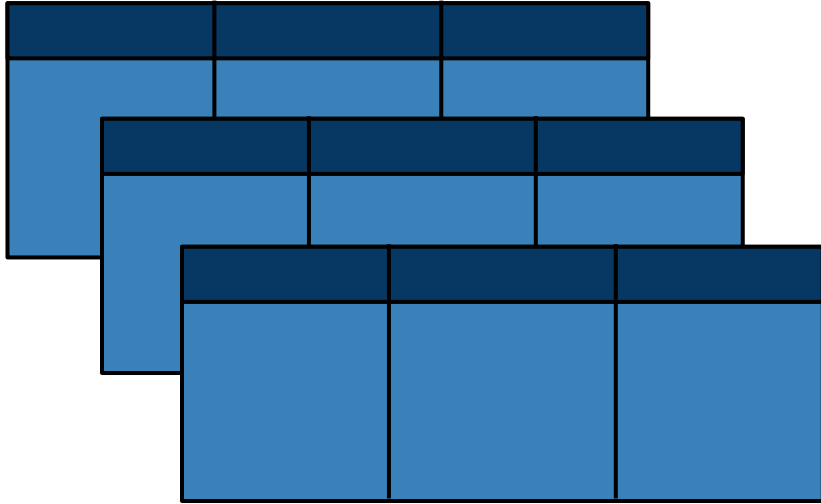
|-- A: string (nullable = true)
|-- B: string (nullable = true)
|-- C: string (nullable = true)

randomSplit

weights = [0.8,0.2]

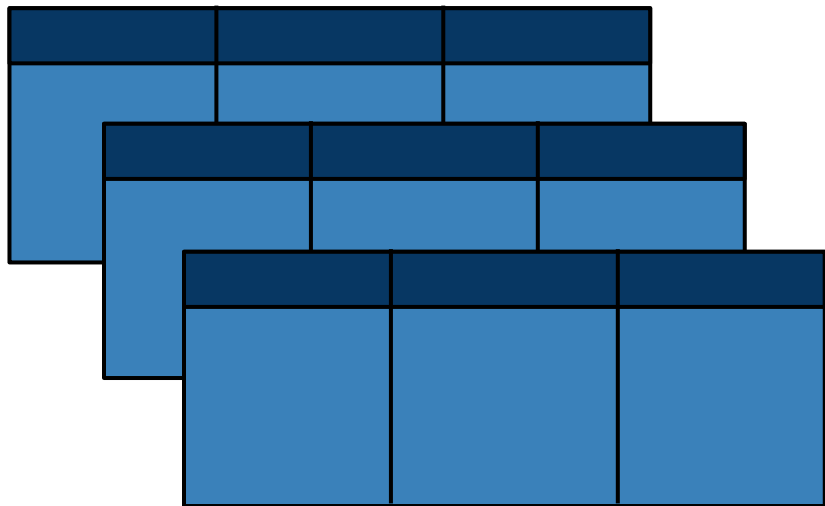


rdd

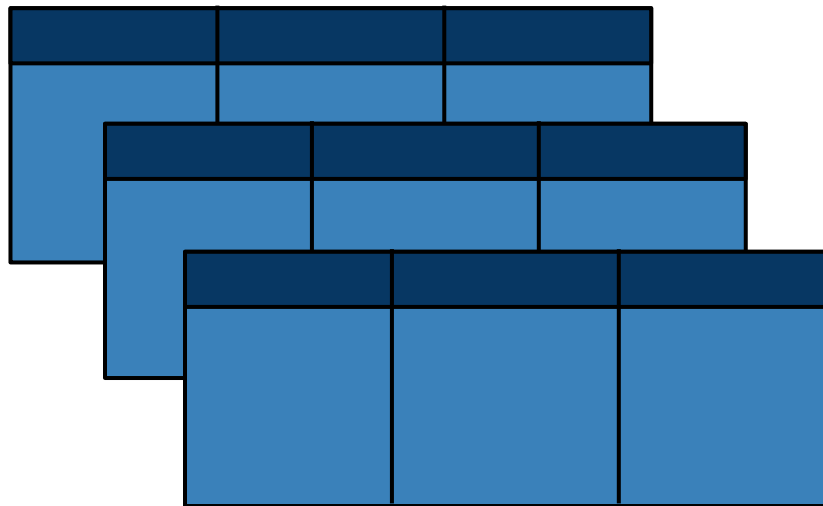


registerTempTable

name = "myTable"



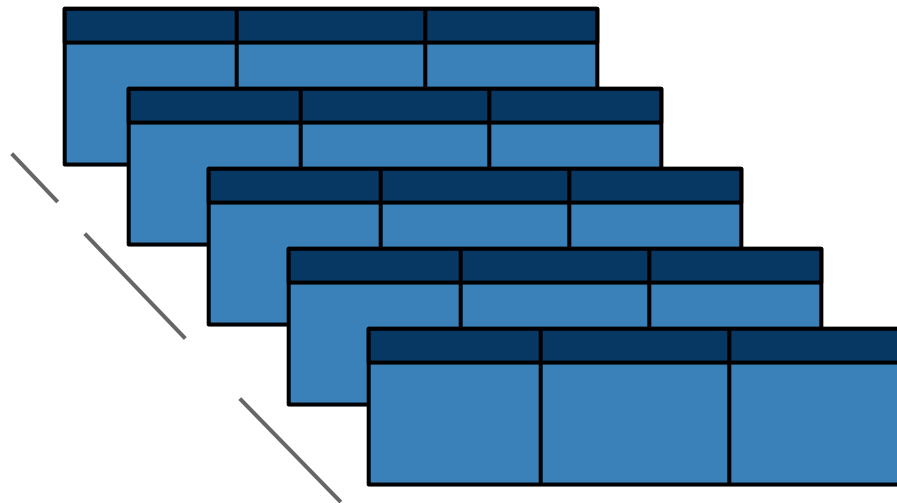
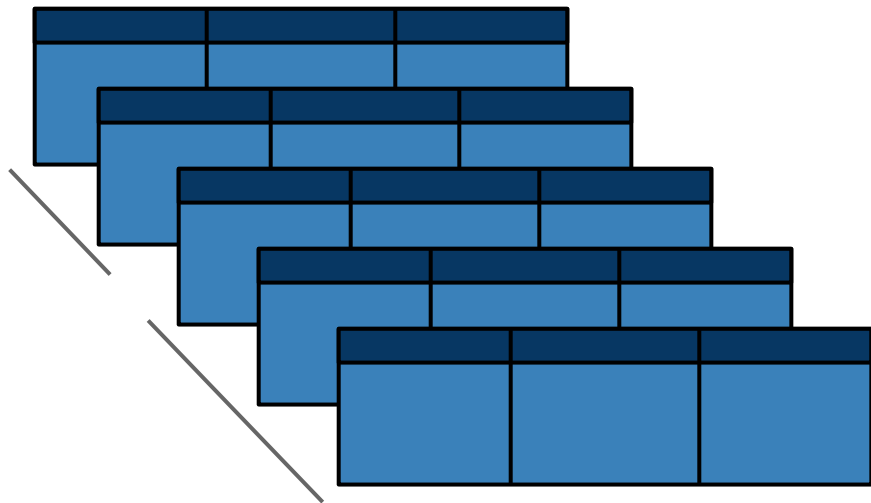
A diagram showing a 3x3 grid of table blocks. Each block is a rectangle divided into a dark blue header and a light blue body. The blocks are arranged in a staggered, overlapping fashion, with each block shifted one unit to the right and one unit down relative to the one above it.



A diagram showing a 3x3 grid of table blocks, identical in structure to the one on the left. Each block is a rectangle divided into a dark blue header and a light blue body. The blocks are arranged in a staggered, overlapping fashion, with each block shifted one unit to the right and one unit down relative to the one above it.

repartition

numPartitions = 3



replace

`to_replace = ['foo', 'null'] value = ['foo-bar', 'unknown']`

foo	null	foo
	null	foo
		baz
A	B	C
bar	foo	null

foo-bar	unknown	foo-bar
	unknown	foo-bar
		baz
A	B	C
bar	foo-bar	unknown

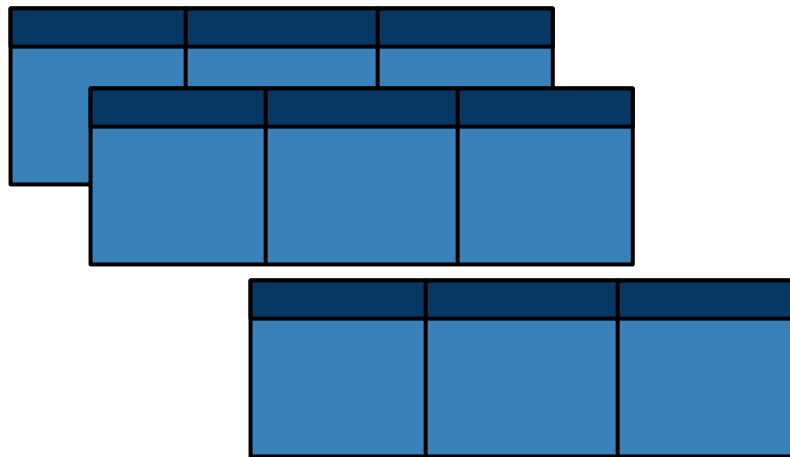
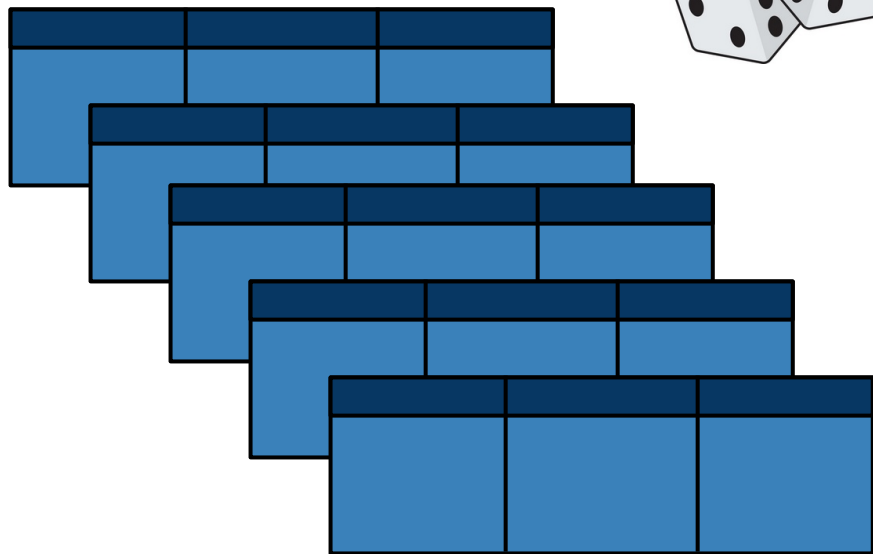
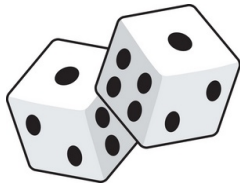
rollup

cols = [A,C]

	cols = [A,C]		
	A	B	C
1	2	20	200
10			
100			

null	null			
1	null			
2	null			
2	200			
A	C	agg(A)	agg(B)	agg(C)
1	100			

sample



schema

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

A	B	C
---	---	---

select

cols = ['B', 'C']

foo	foo	1
bar	foo	2
A	B	C
bar	foo	3

foo	1
foo	2
B	C
foo	3

selectExpr

expr = ["substr(A,1,1)", "C + 10"]

foo	foo	1
bar	foo	2
A	B	C
bar	foo	3

f	11
b	12
A	C
b	13

show

foo	3	0.2

bar	2	1.0

A	B	C
bar	1	2.2

stdout

```
+---+---+
| A|B| C|
+---+---+
|foo|3|0.2|
|bar|2|1.0|
|bar|1|2.2|
+---+---+
```

sort

cols = ['A', 'C'], ascending = [True, False]

baz	baz	3
bar	foo	1
A	B	C
baz	foo	2

baz	foo	2
baz	baz	3
A	B	C
bar	foo	1

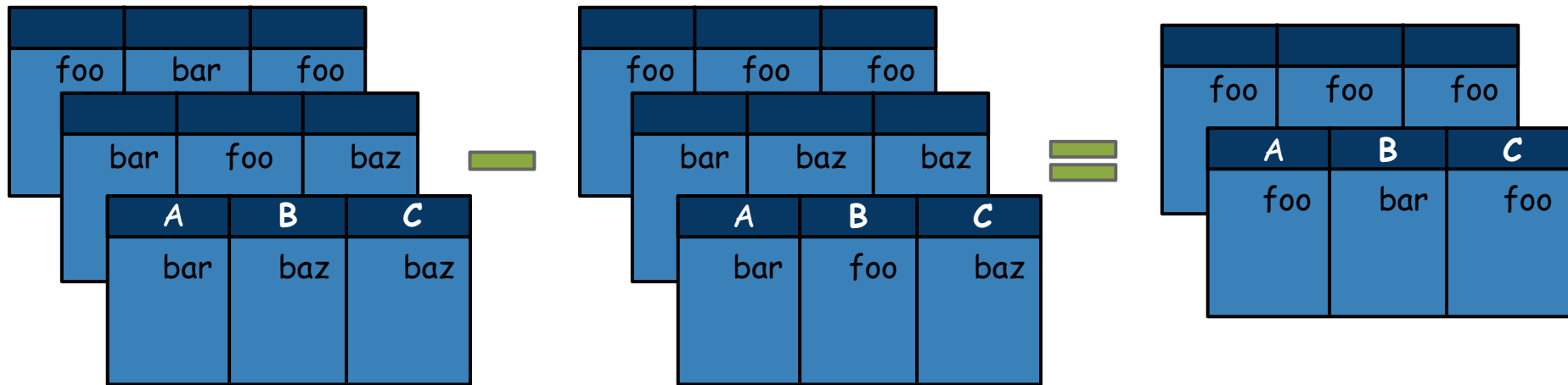
stat

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

subtract



take

$n = 2$

foo	foo	foo
bar	bar	baz
A	B	C
bar	foo	baz

[Row(A='bar', B='foo', C='baz'),
Row(A='bar', B='bar', C='baz')]

toJSON

foo	3	0.2

bar	2	1.0

A	B	C
bar	1	2.2

```
u'{"A":"foo","B":3,"C":0.2}'
```

```
u'{"A":"bar","B":2,"C":1.0}'
```

```
u'{"A":"bar","B":1,"C":2.2}'
```

toPandas

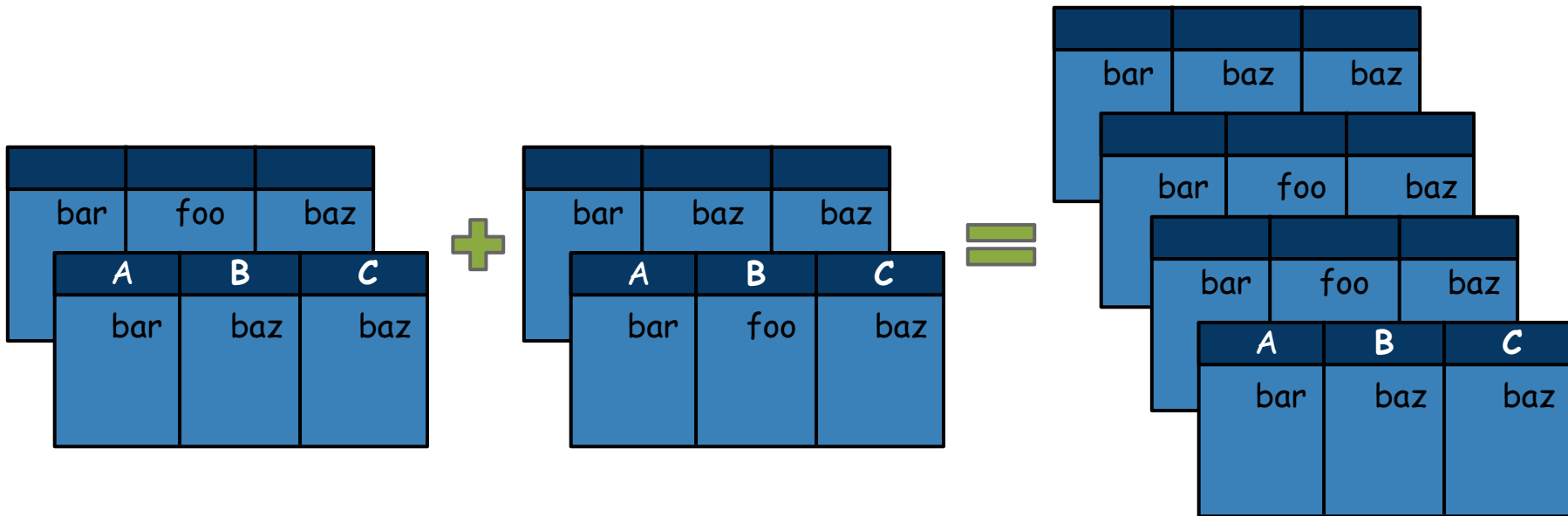
foo	3	0.2

bar	2	1.0

A	B	C
bar	1	2.2

	A	B	C
0	foo	3	0.2
1	bar	2	1.0
2	bar	1	2.2

unionAll



unpersist

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null

where

condition = "A = foo"

foo	foo	foo

bar	foo	baz

A	B	C
bar	foo	baz

A	B	C
foo	foo	foo

withColumn

colName = 'D'

A	B	C

A	B	C	D

withColumnRenamed

existing = 'C' col = 'D'

A	B	C

A	B	D

write

foo	null	foo

null	foo	baz

A	B	C
bar	foo	null