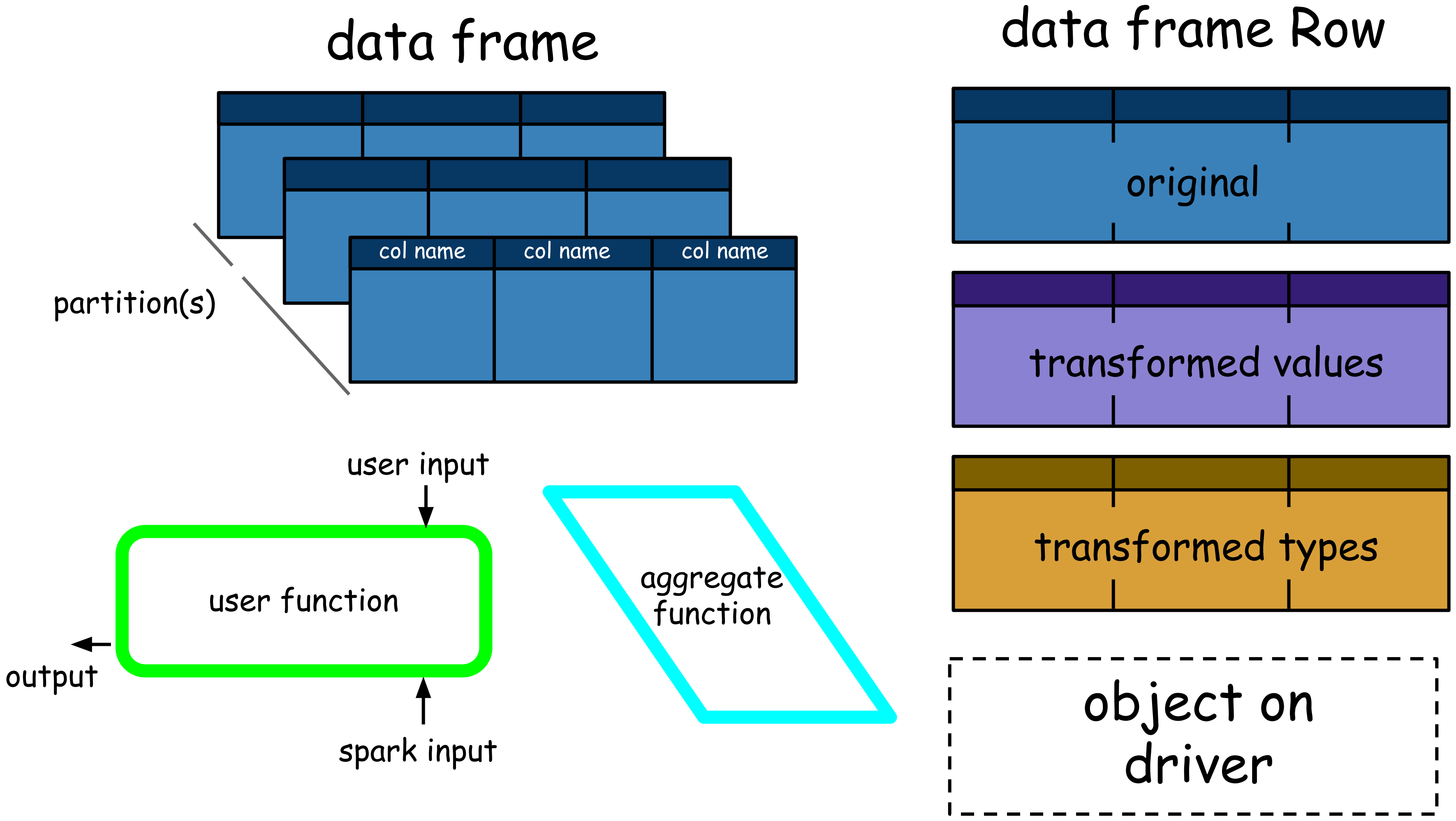
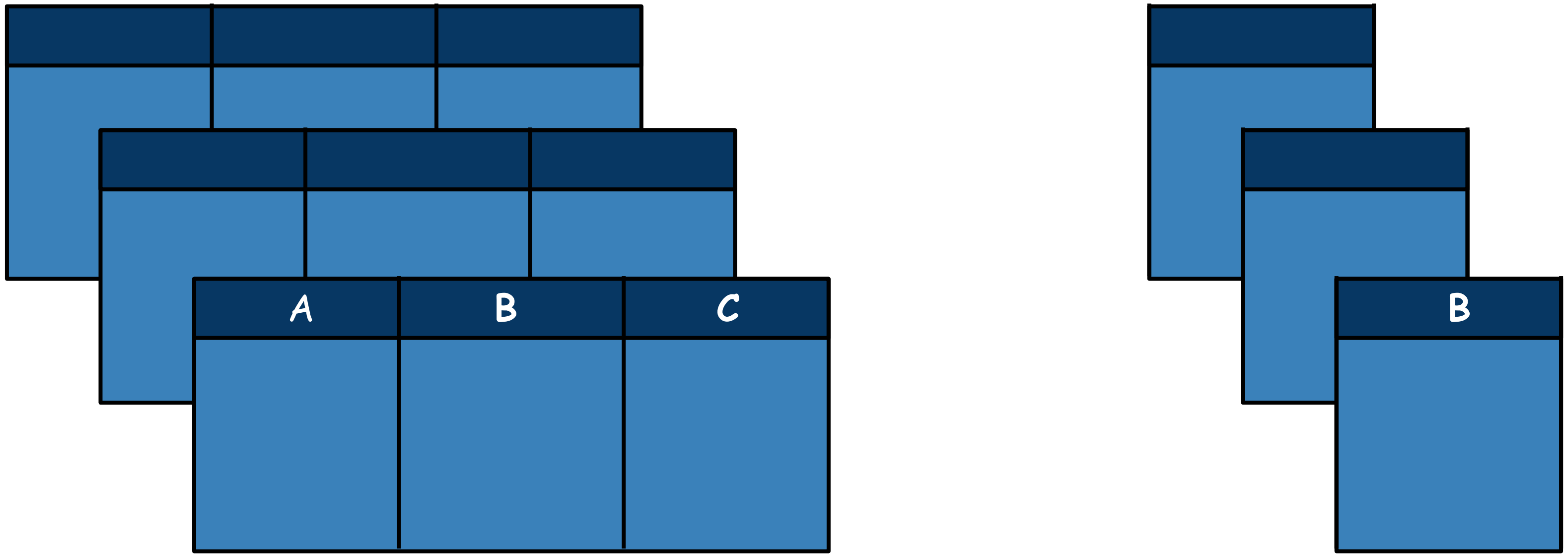
pyspark-pictures data frames

Learn the pyspark API through pictures and simple examples

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

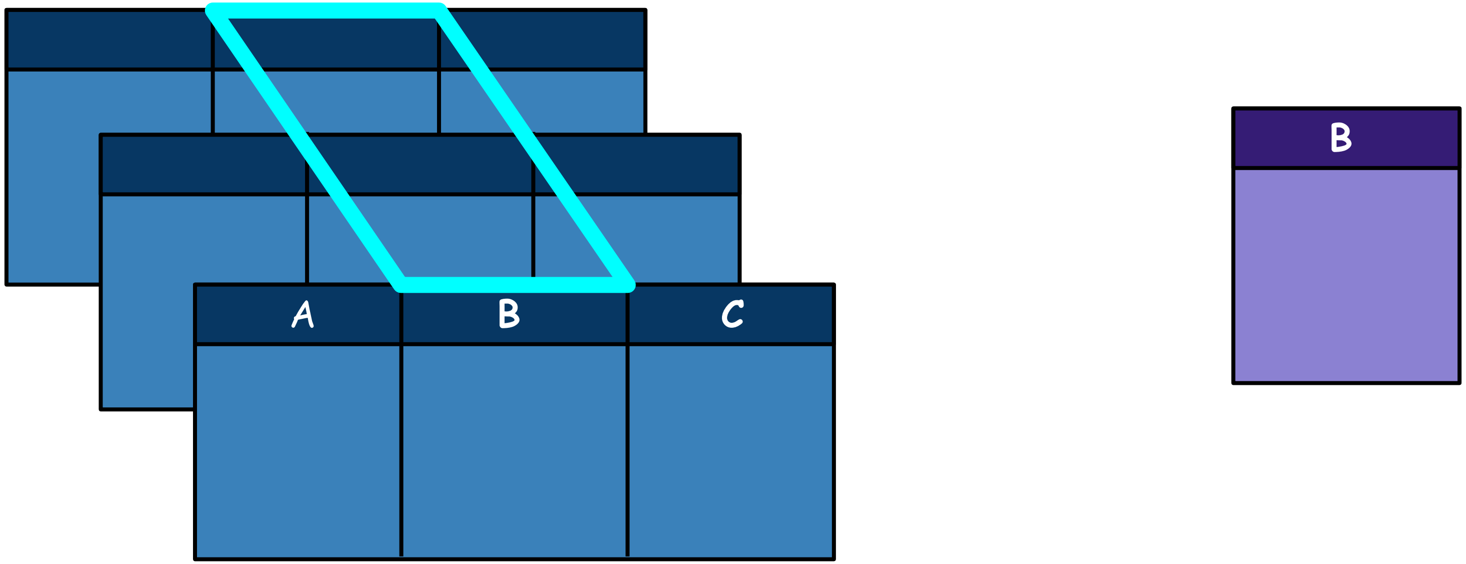

data frame
col name
col name
col name
partition(s)
data frame Row
original
transformed values
transformed types
user input
user function
output
spark input
aggregate function
object on driver

- df.B

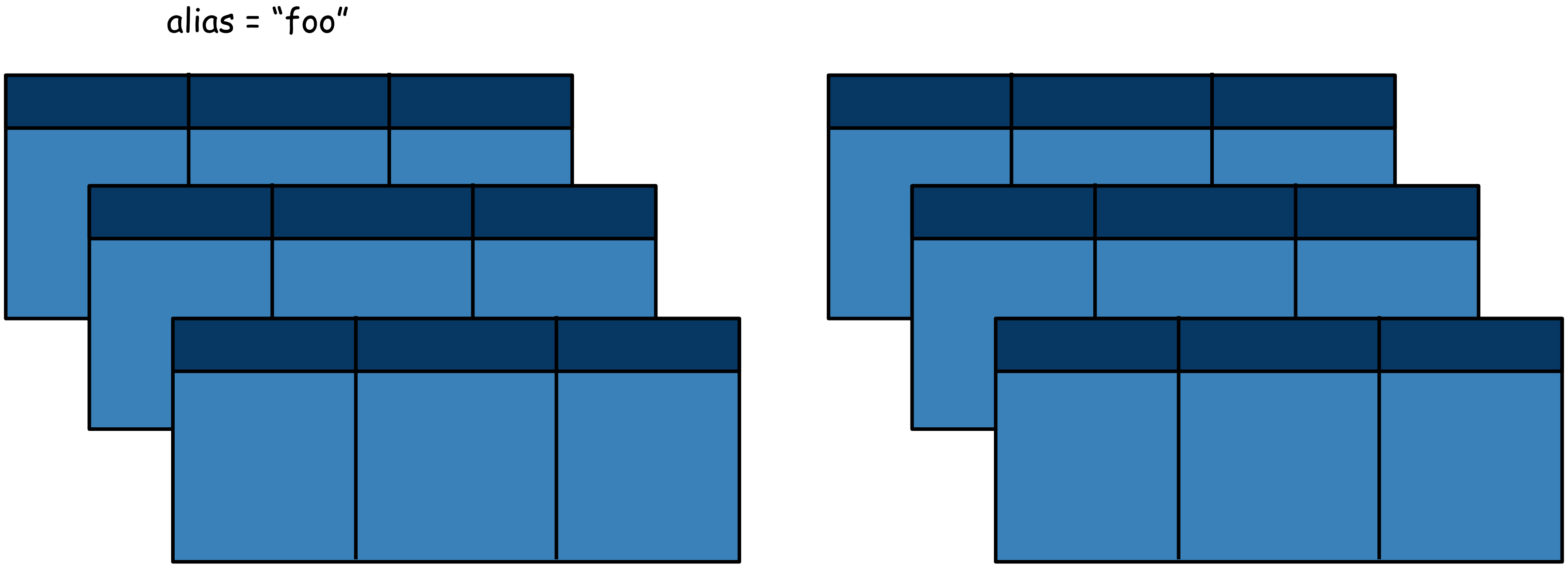


agg

exprs = {"B" : "aggF"}

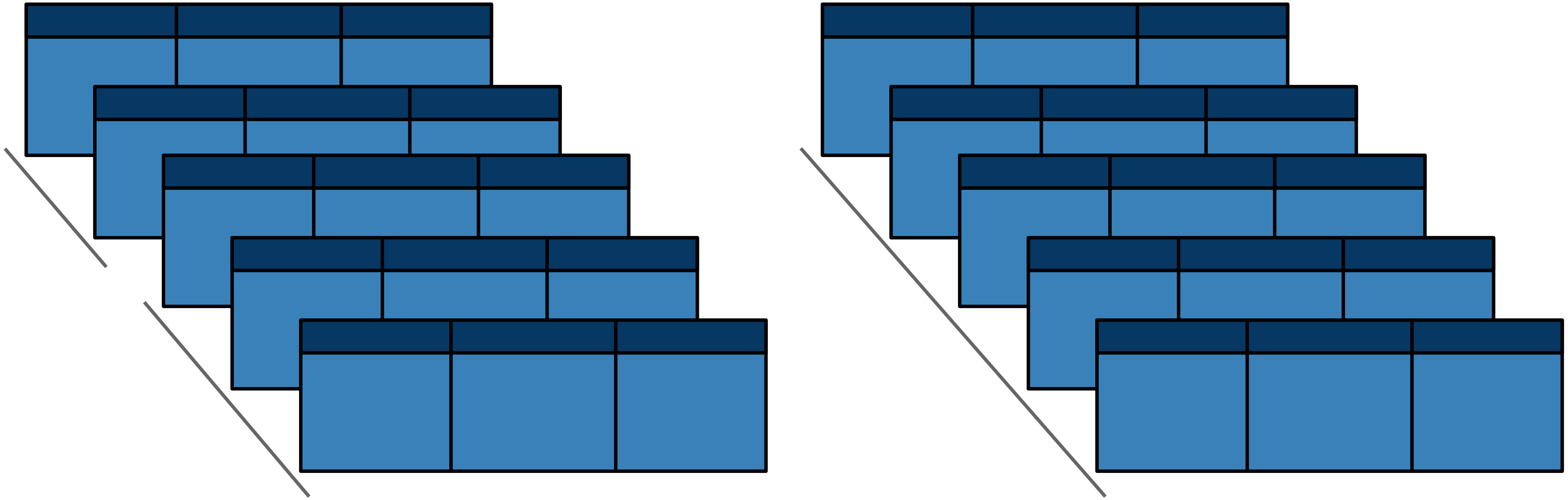


alias

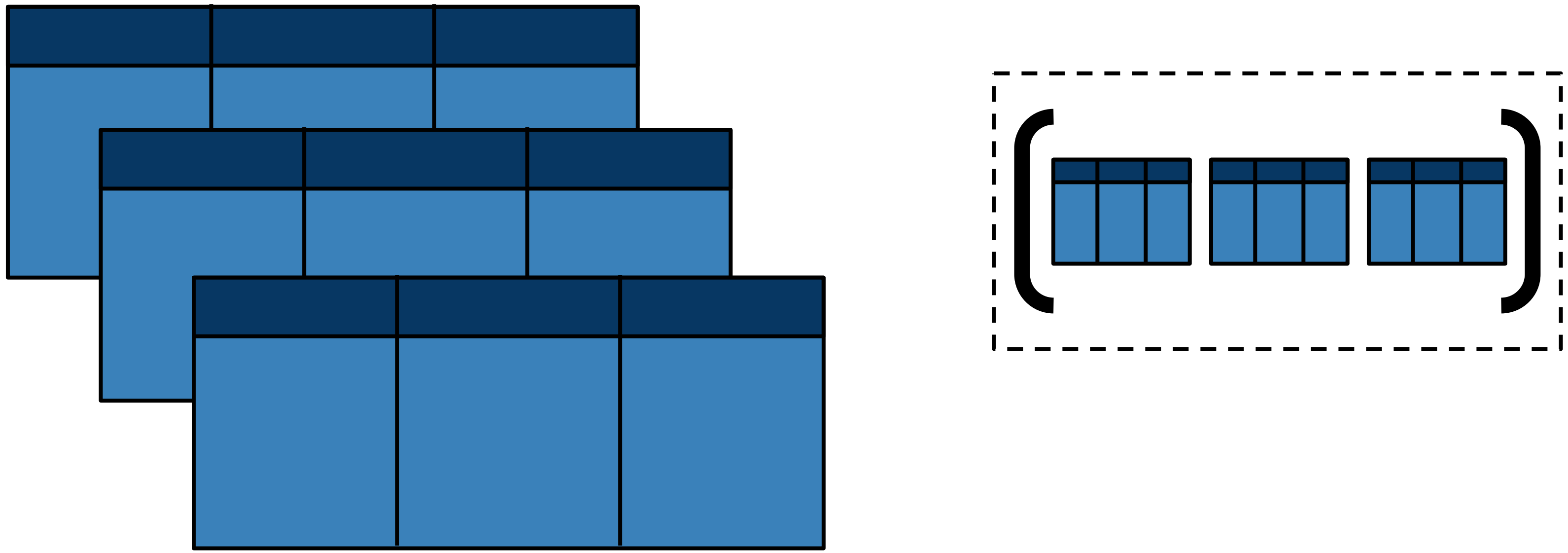


coalesce

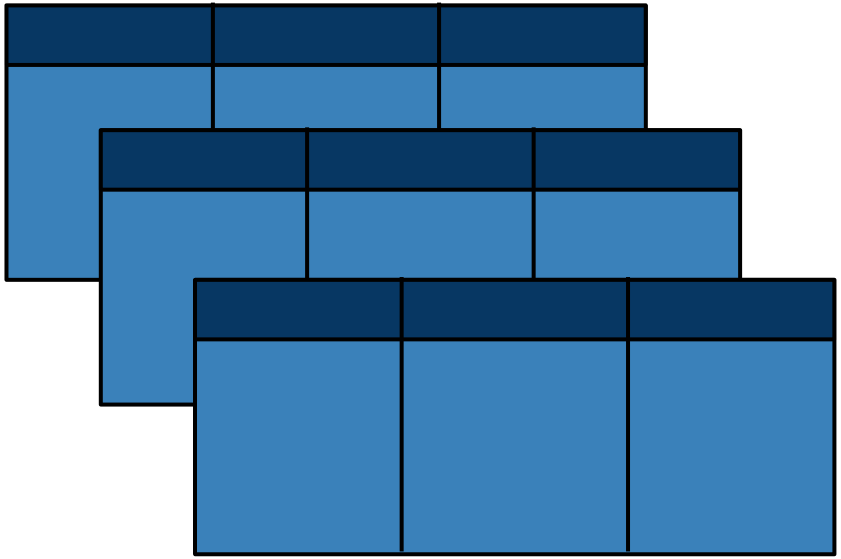
numPartitions = 1

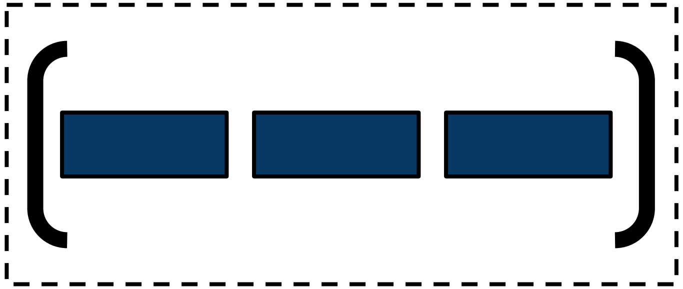


collect

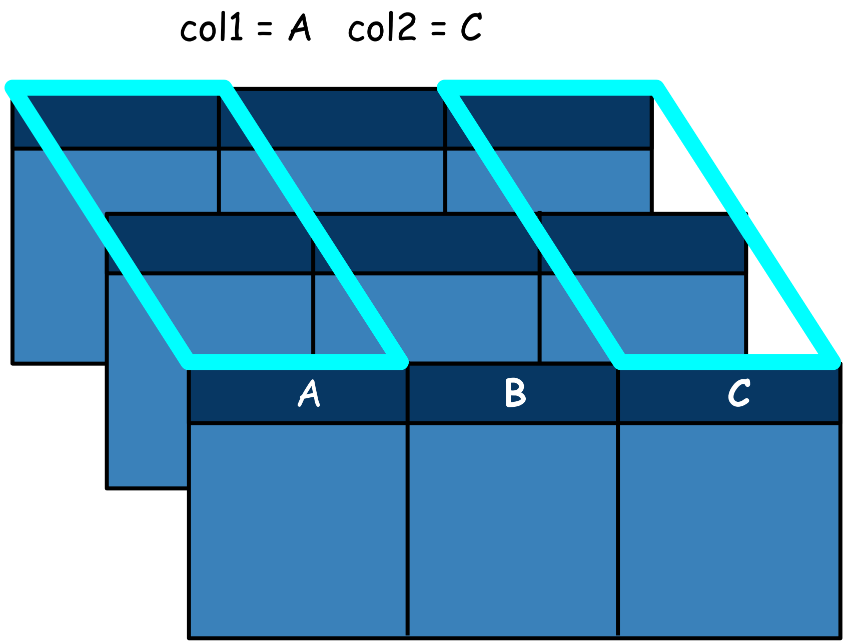


columns





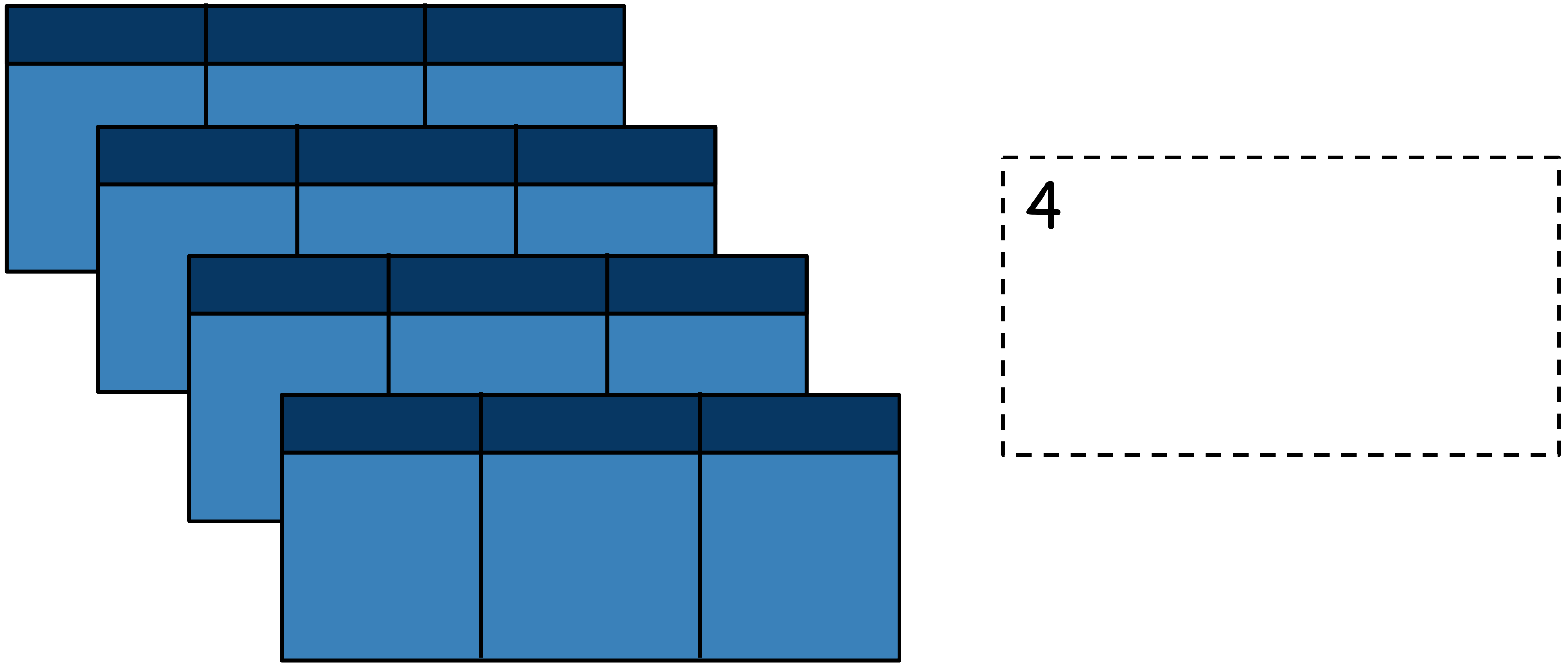
corr



Pearson's r

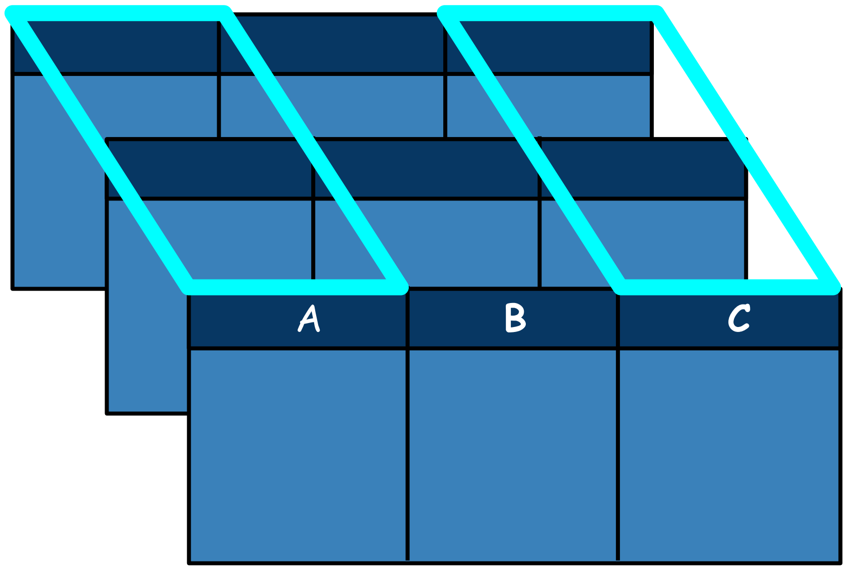
$$r = \frac{\sum_i (A_i - \bar{A})(C_i - \bar{C})}{\sqrt{\sum_i (A_i - \bar{A})^2}\sqrt{\sum_i (C_i - \bar{C})^2}}$$

count



cov

col1 = A   col2 = C

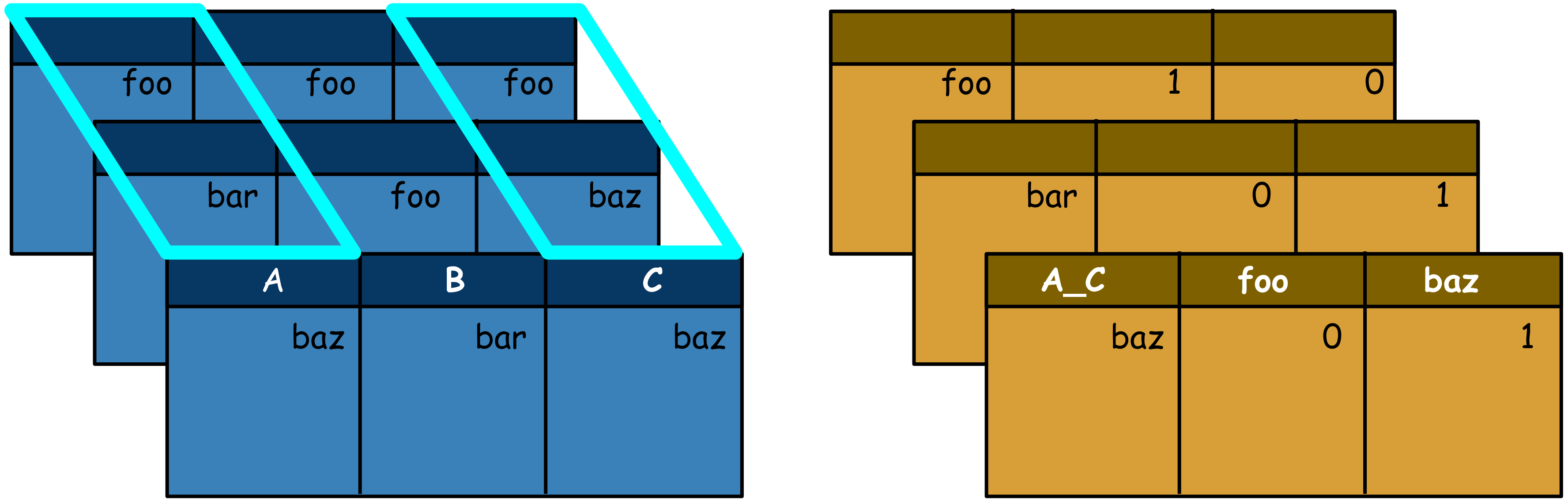


Sample Covariance

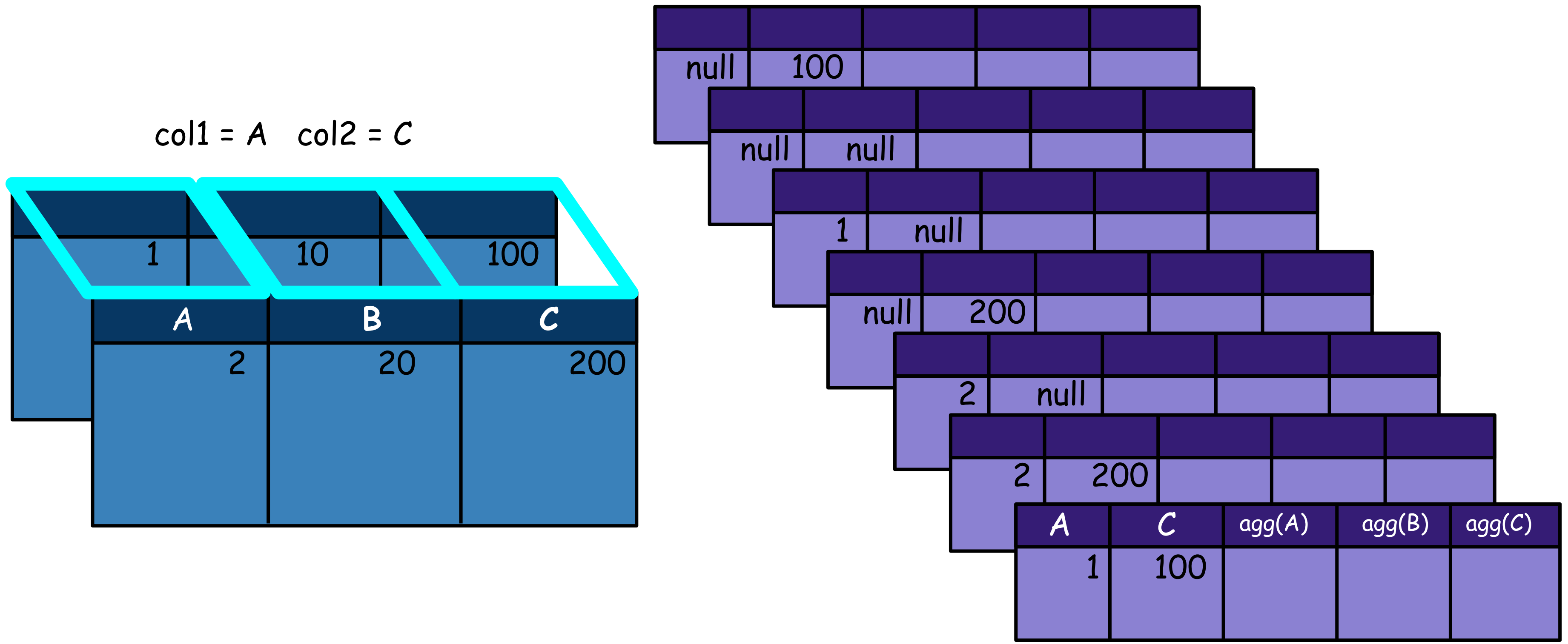
$$\frac{1}{N-1}\sum_i (A_i - \bar{A})(C_i - \bar{C})$$

crosstab

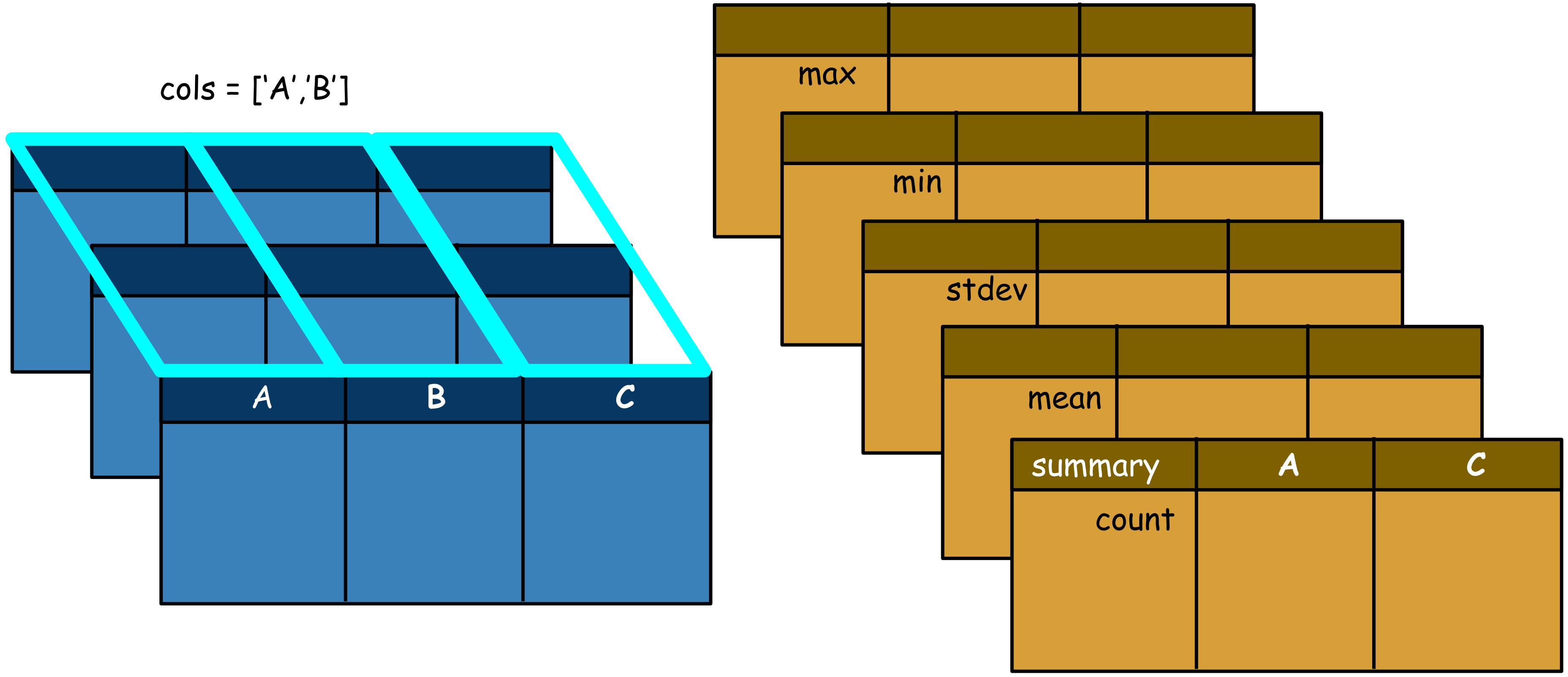
col1 = A   col2 = C



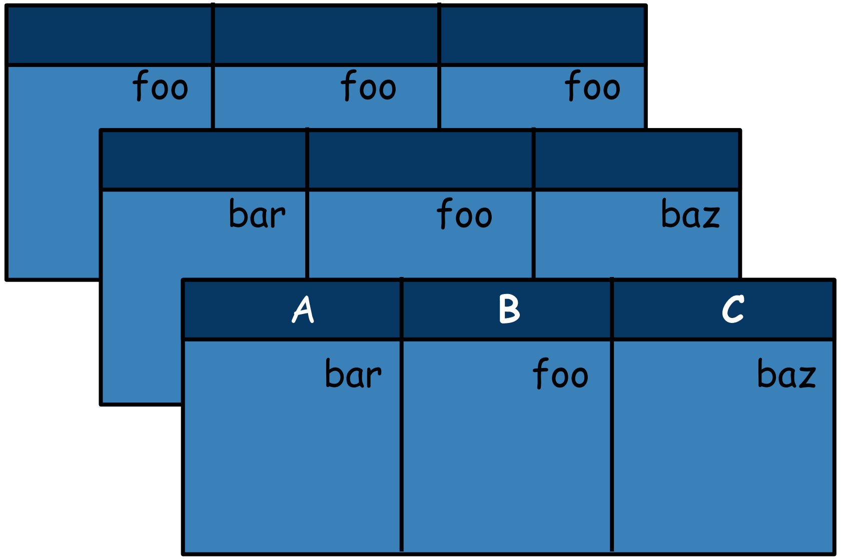
cube


col1 = A
col2 = C
A
B
C
1
10
100
2
20
200
null 100
null null
1 null
null 200
2 null
2 200
A
C
agg(A)
agg(B)
agg(C)
1 100

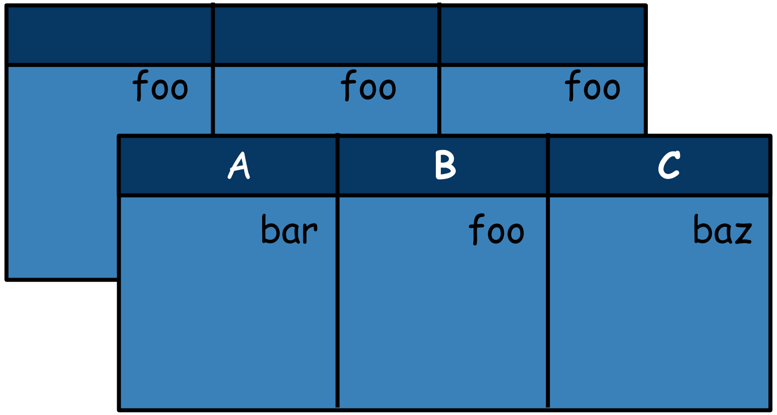
describe



distinct

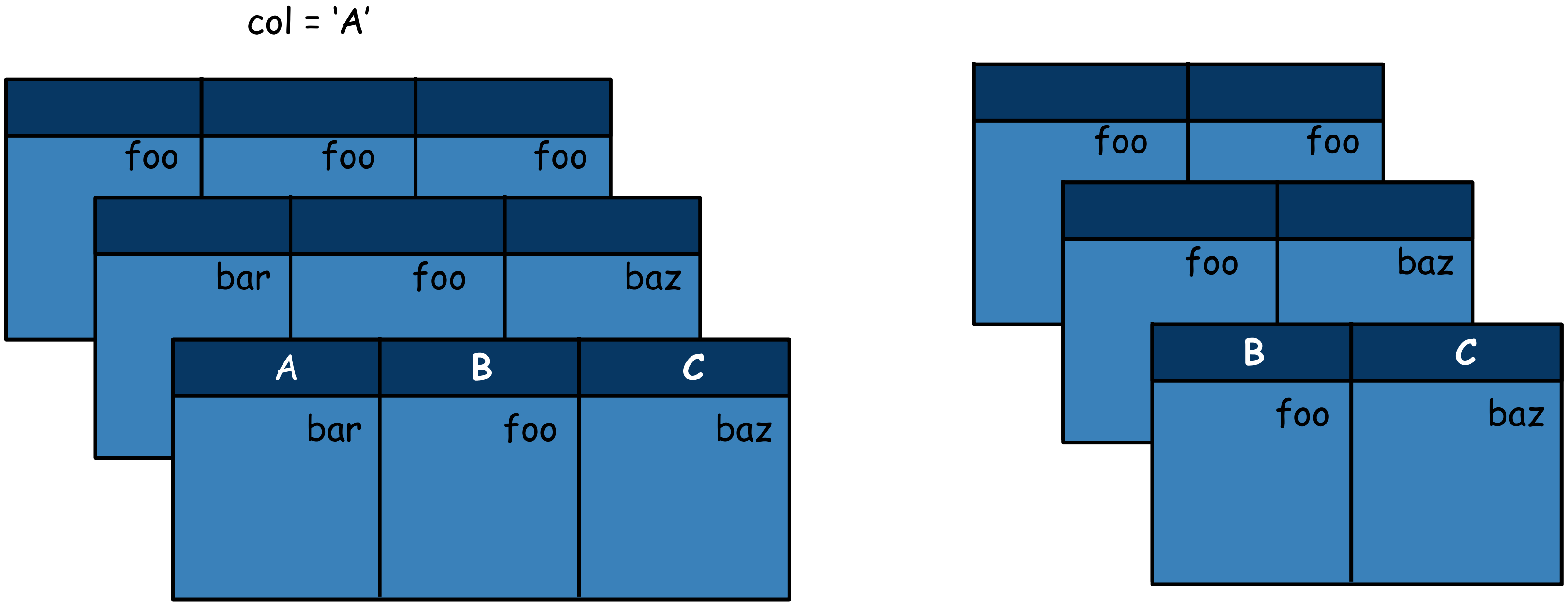


foo
foo
foo
bar
foo
baz
A
B
C
bar
foo
baz

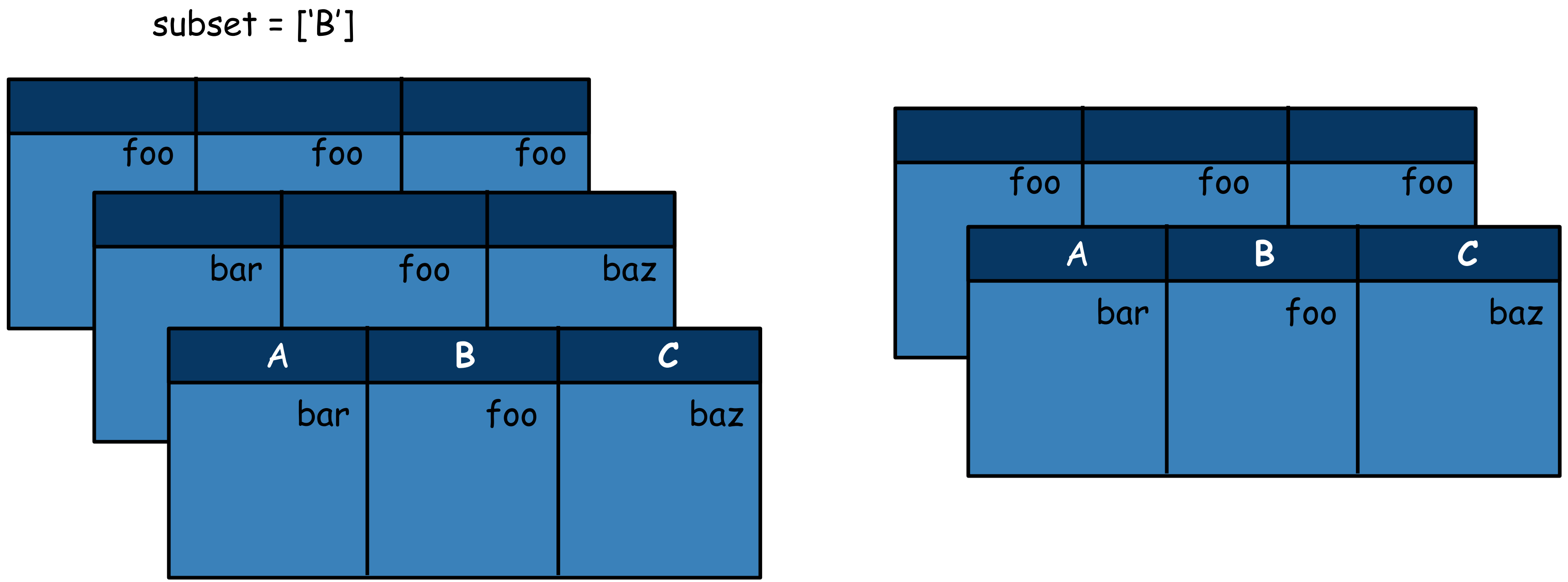


foo
foo
foo
A
B
C
bar
foo
baz

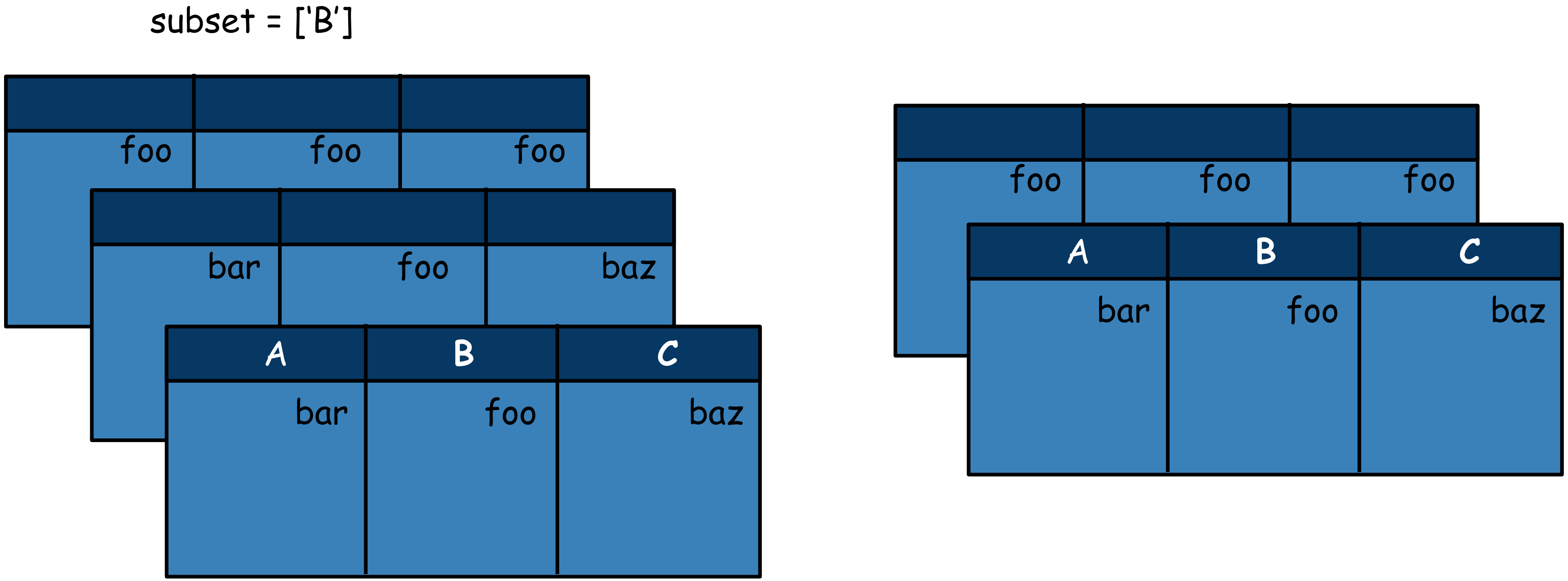
drop


col = 'A'
foo
foo
foo
bar
foo
baz
A
B
C
bar
foo
baz
foo
foo
foo
baz
B
C
foo
baz

dropDuplicates

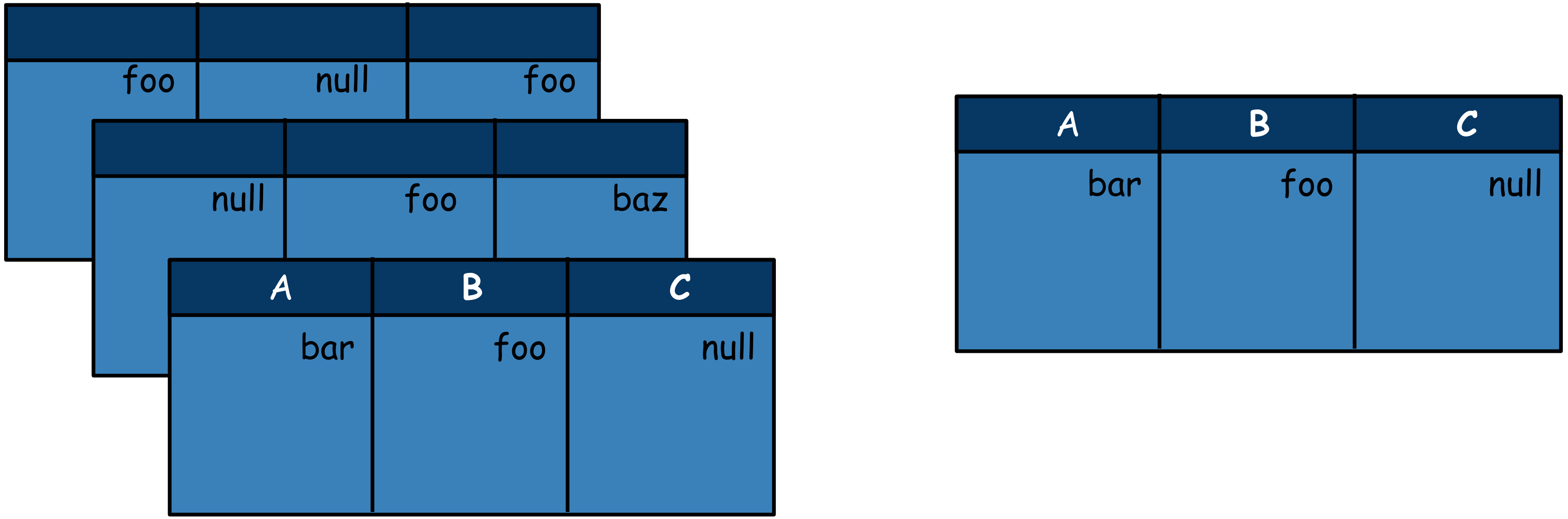


drop_duplicates

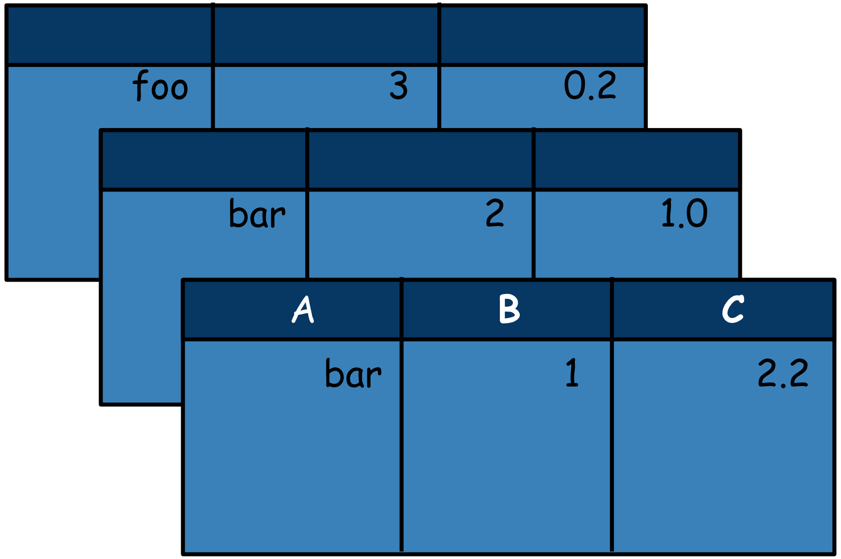


dropna

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



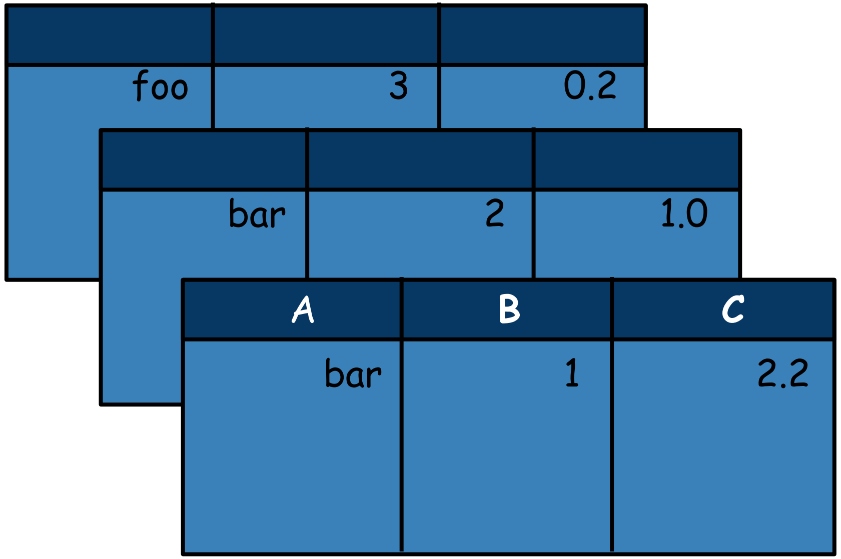
dtypes



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

explain

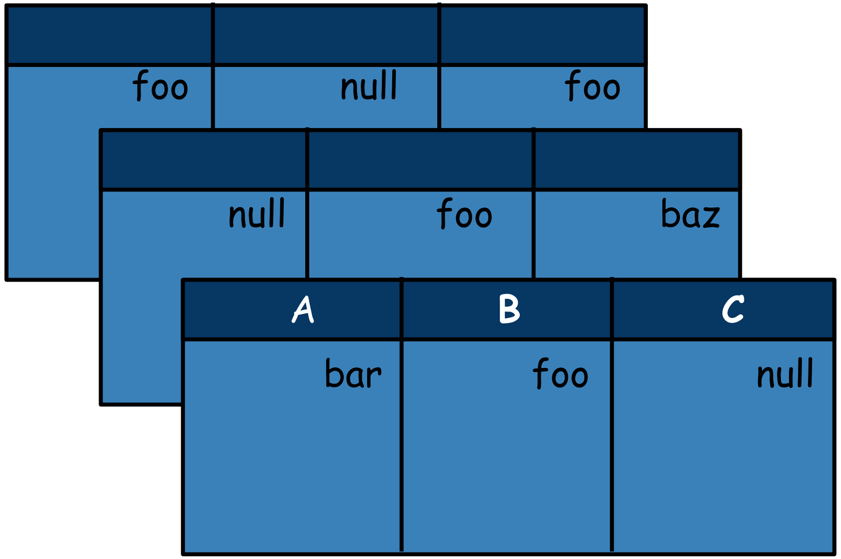
extended = True

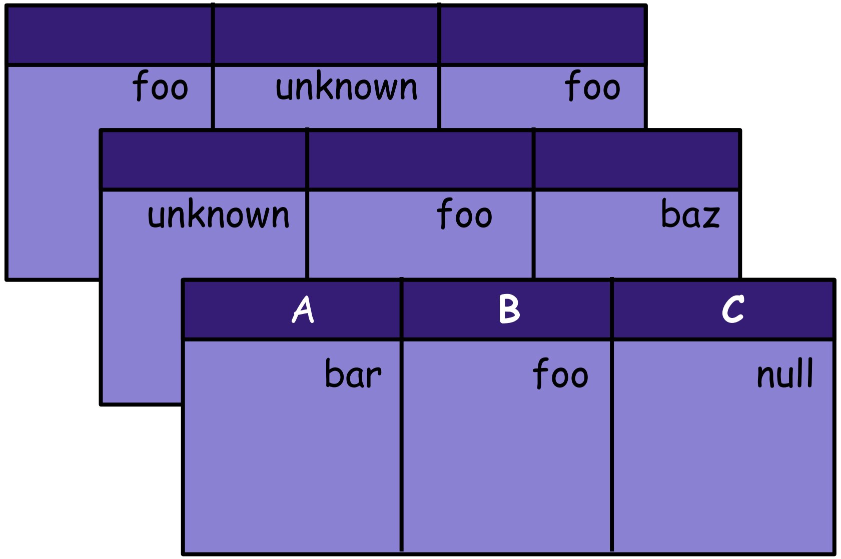


```
== Parsed Logical Plan ==
...
== Analyzed Logical Plan ==
...
== Optimized Logical Plan ==
...
== Physical Plan ==
...
== RDD ==
```

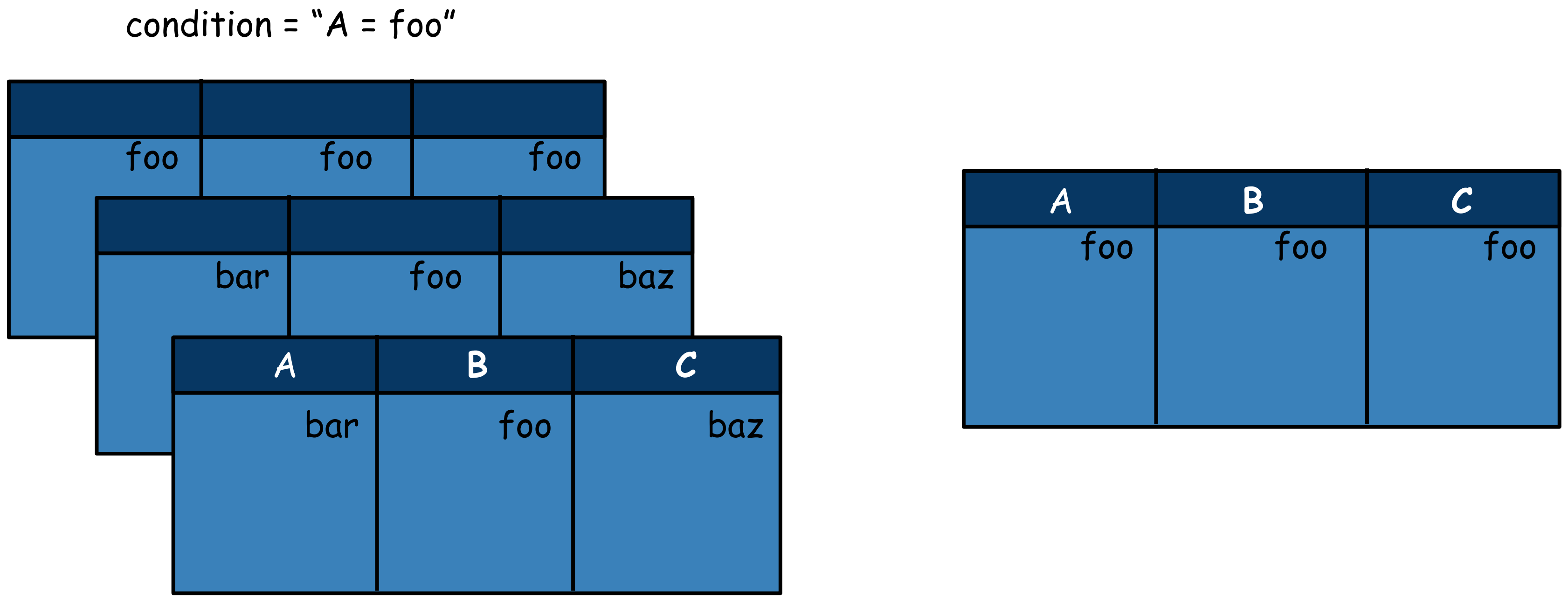
fillna

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

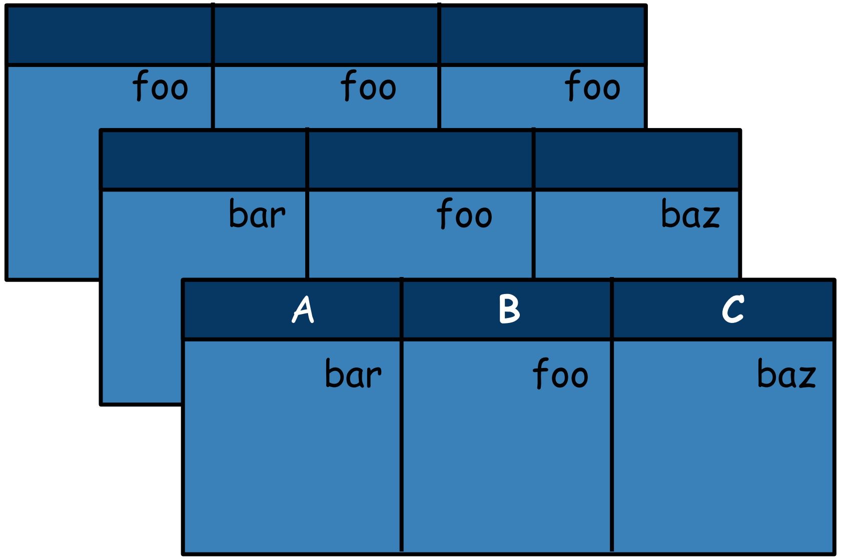




filter

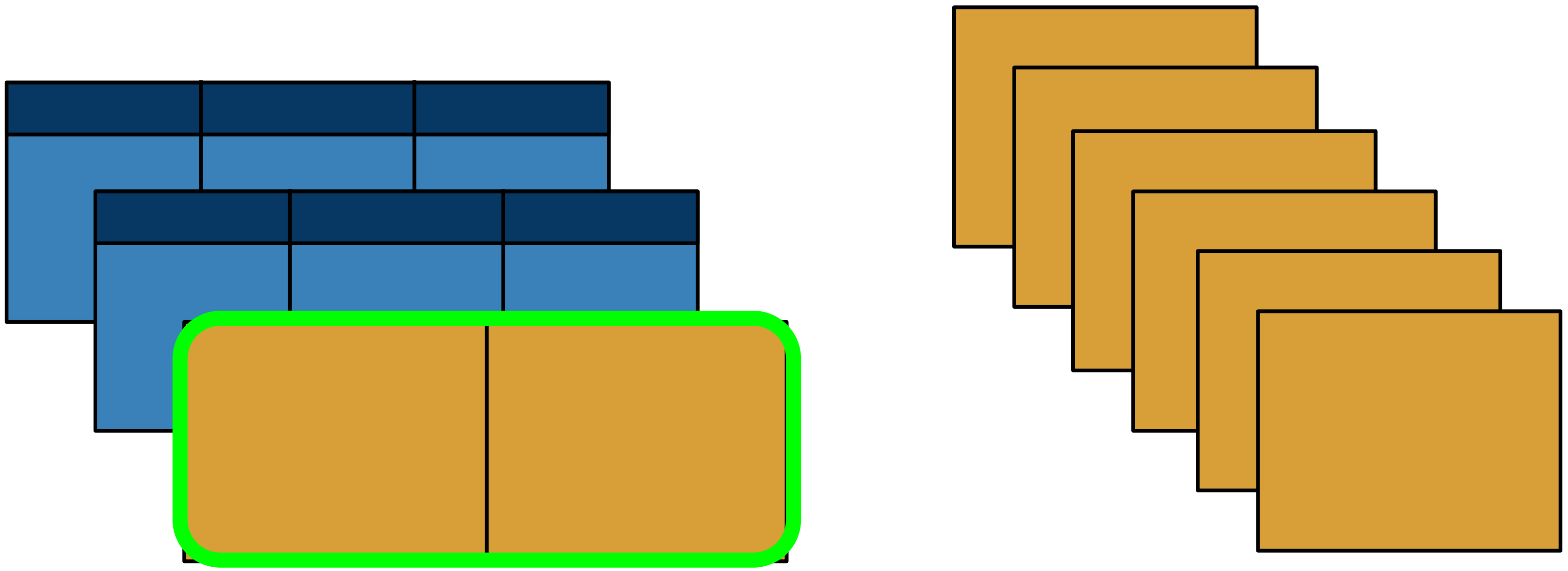

condition = "A = foo"
foo
foo
foo
bar
foo
baz
A
B
C
bar
foo
baz
A
B
C
foo
foo
foo

first

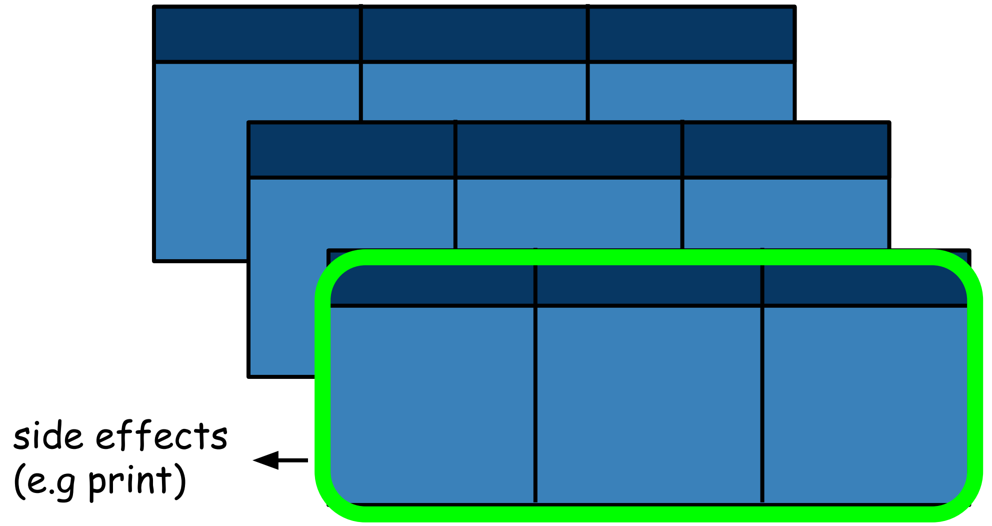


Row(A='bar', B='foo', C='baz')

flatMap

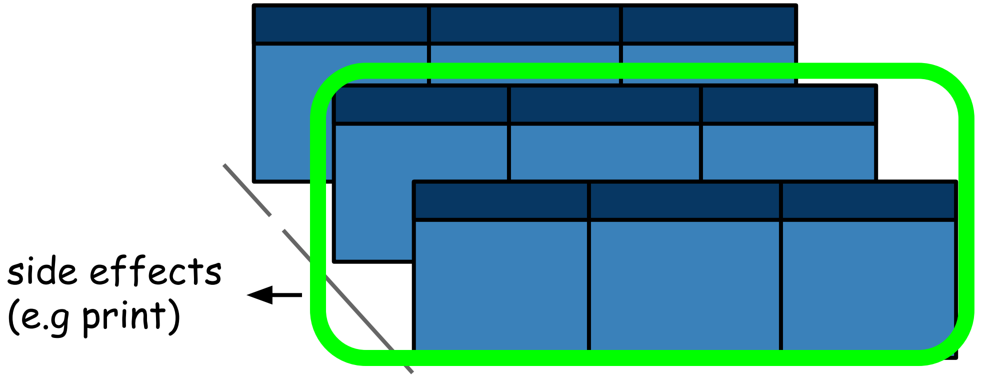


foreach



*no return value, original DataFrame unchanged

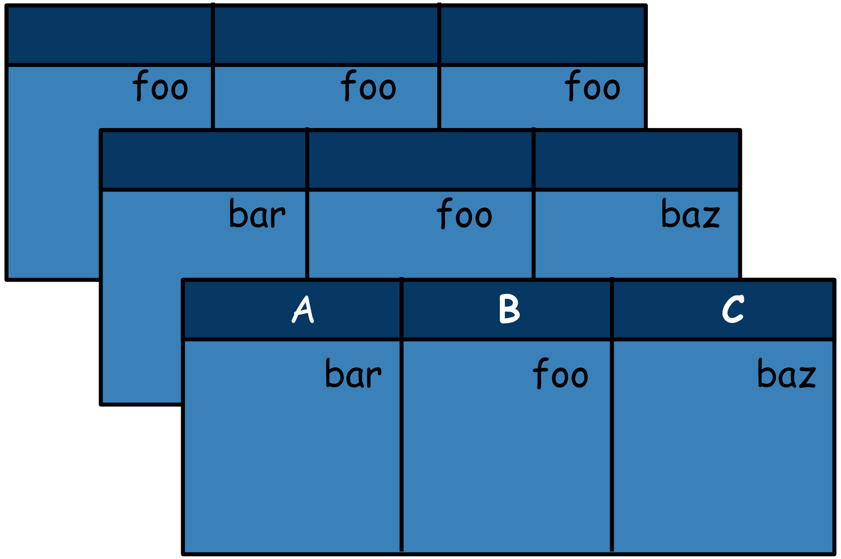
foreachPartition



*no return value, original DataFrame unchanged

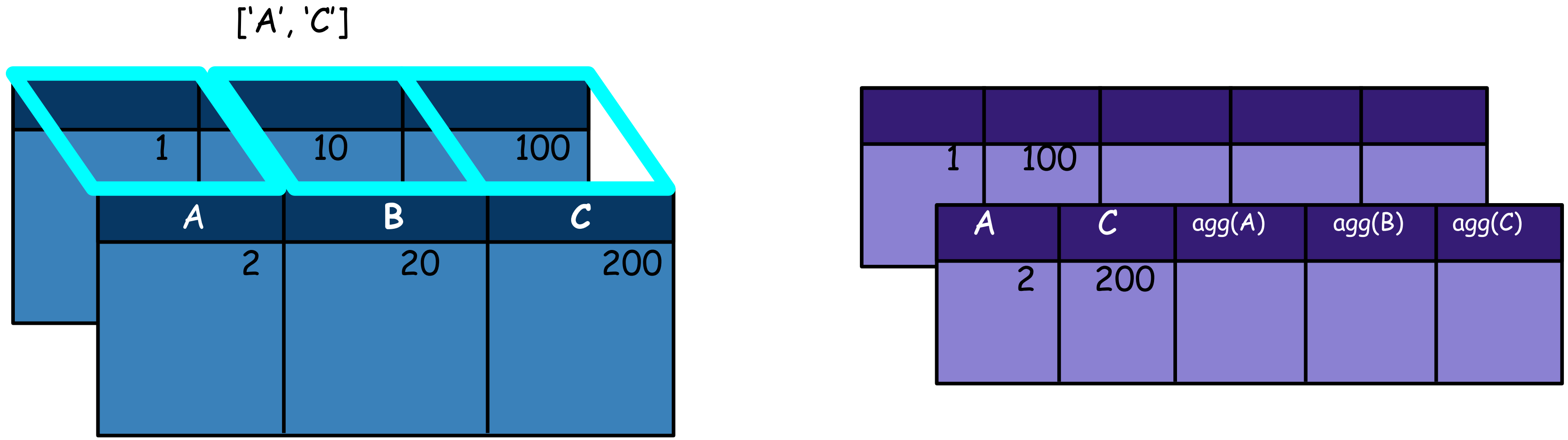
freqItems

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

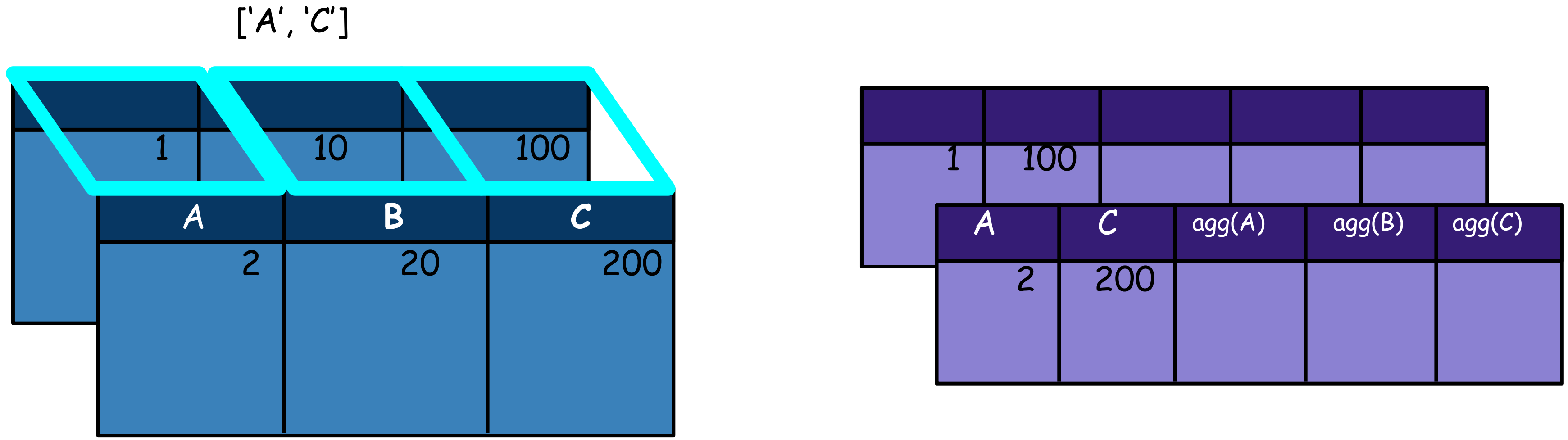


A_freqItems	C_freqItems
[bar,foo]	[baz,foo]

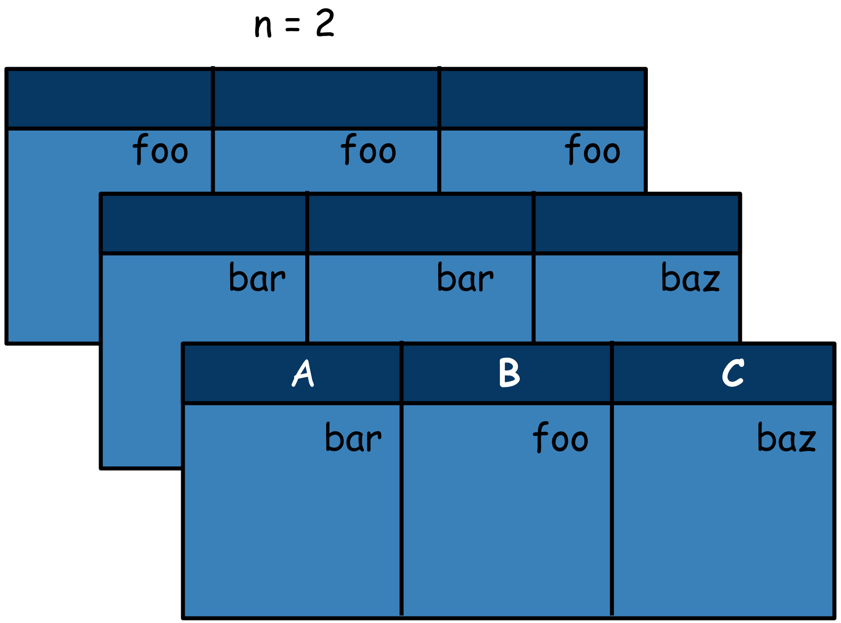
groupBy



groupby

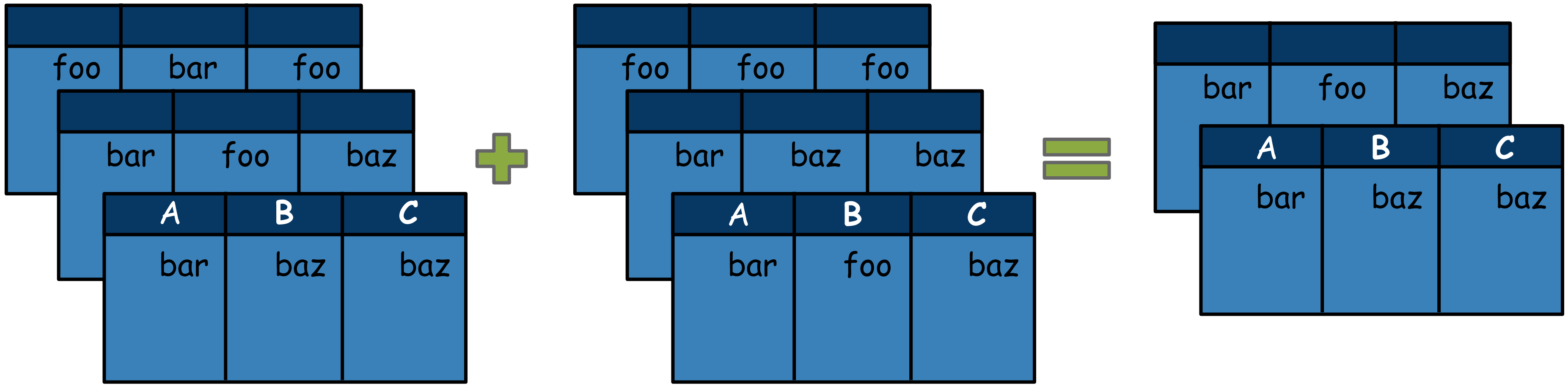


head



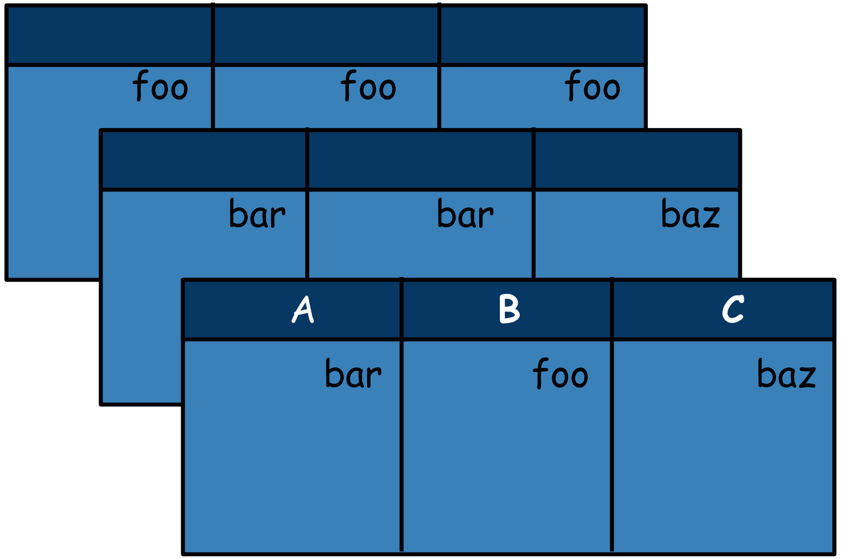
```
[Row(A='bar', B='foo', C='baz'),
Row(A='bar', B='bar', C='baz')]
```

intersect



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

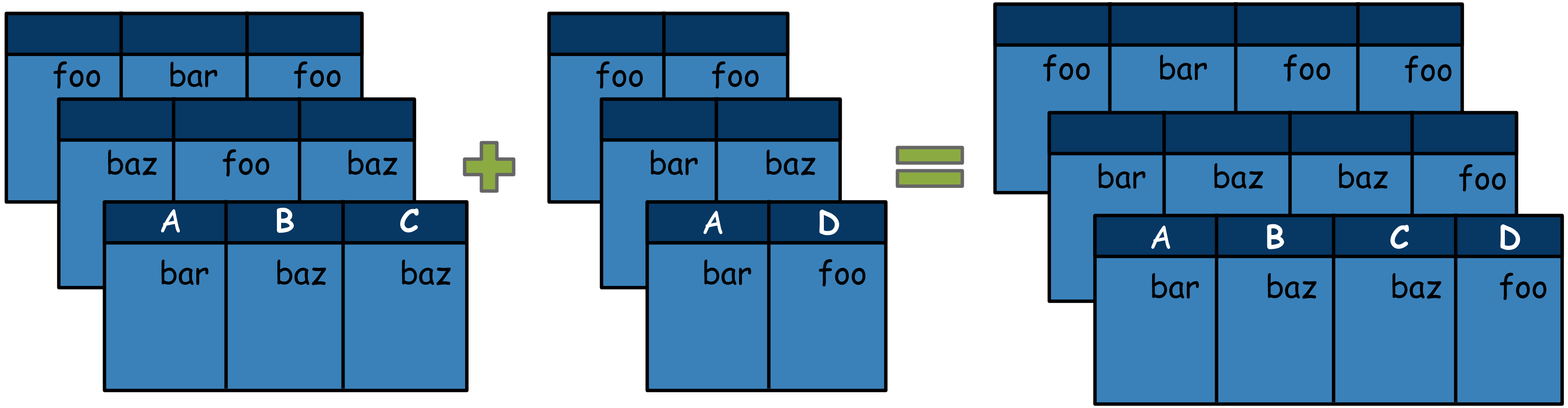
isLocal



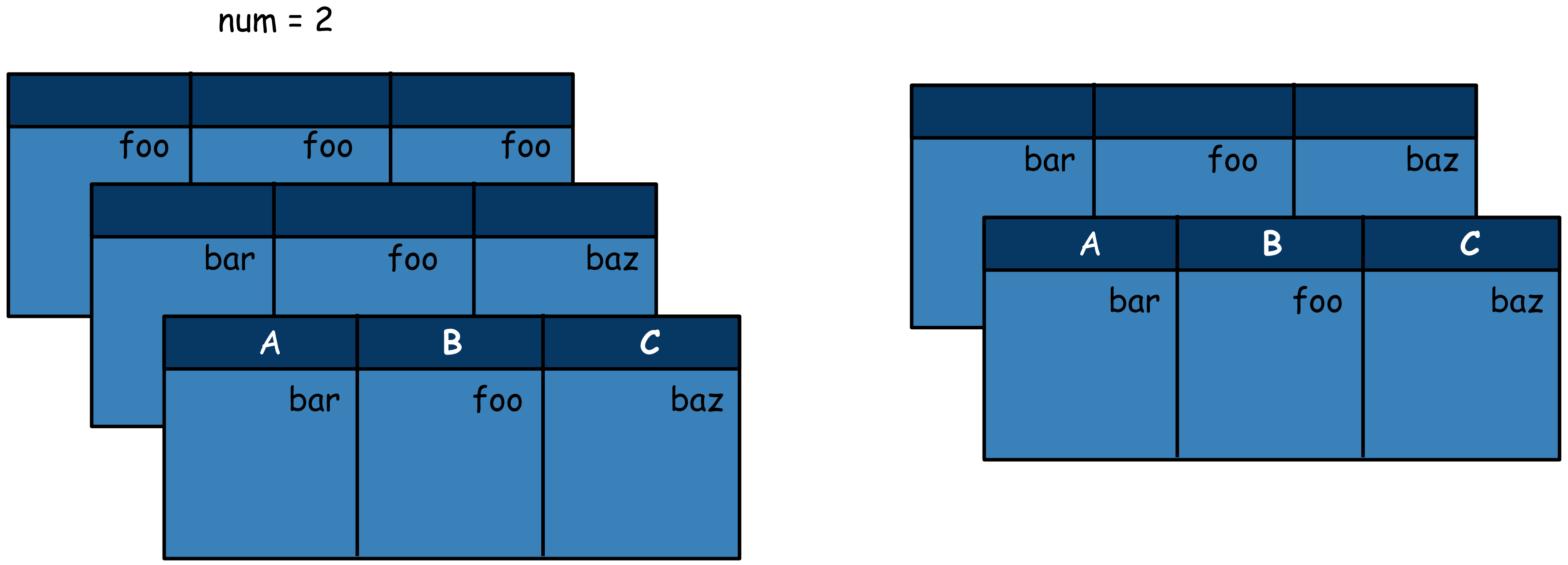
True

join

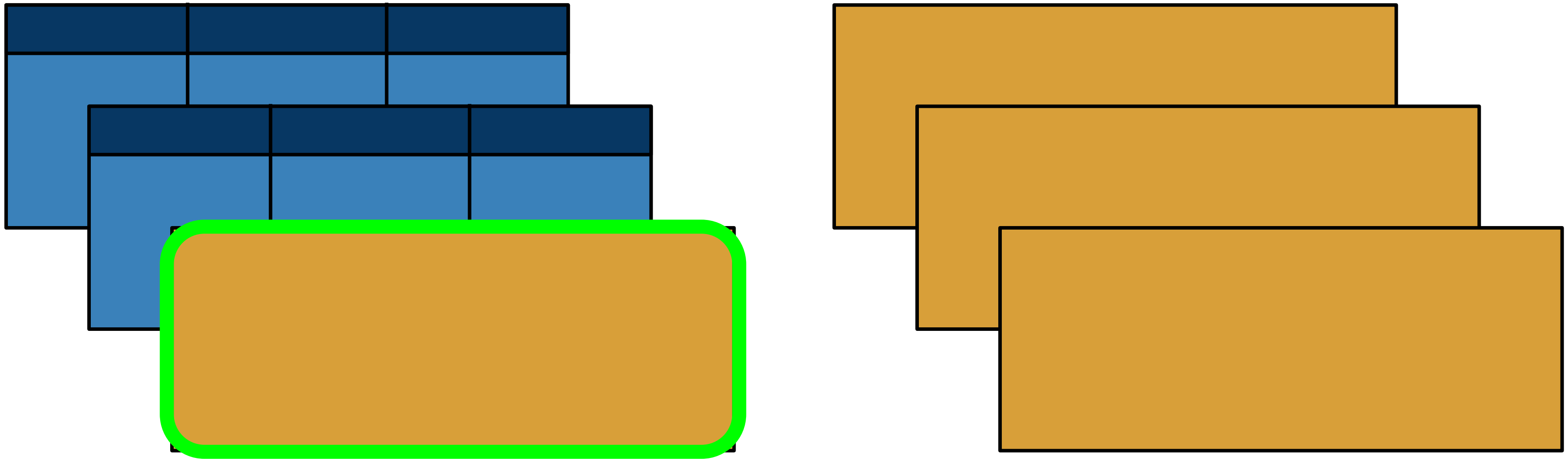
joinExprs = 'A', joinType = 'inner'



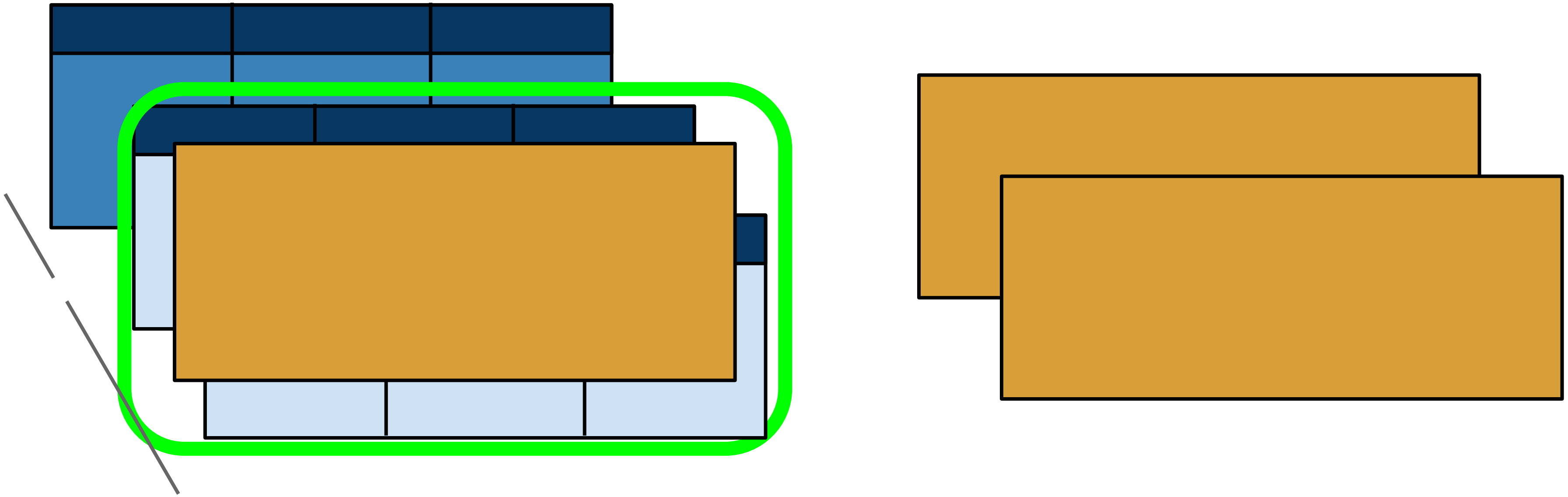
limit


num = 2
foo
foo
foo
bar
foo
baz
A
B
C
bar
foo
baz
bar
foo
baz
A
B
C
bar
foo
baz

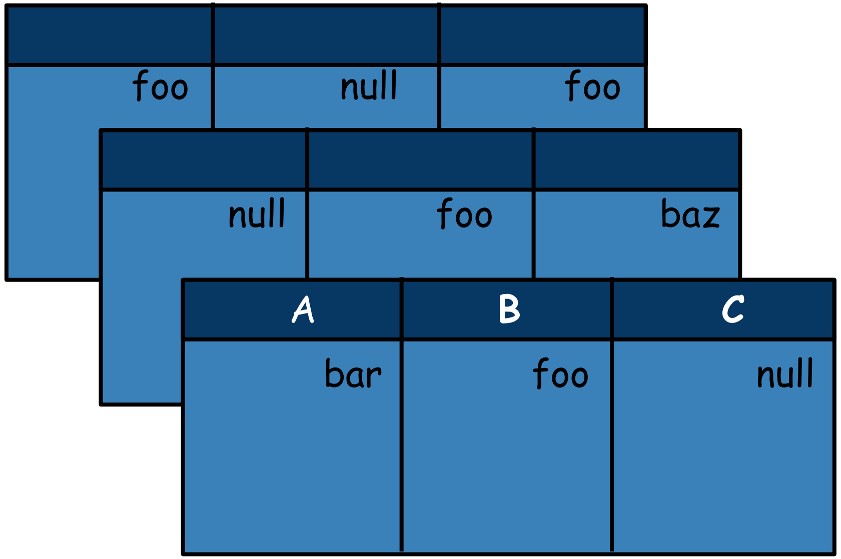
map



mapPartitions

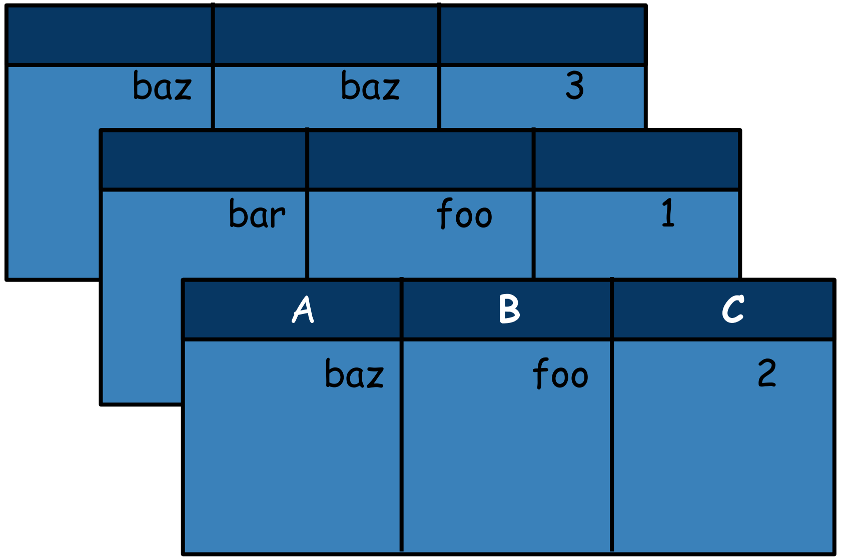


na

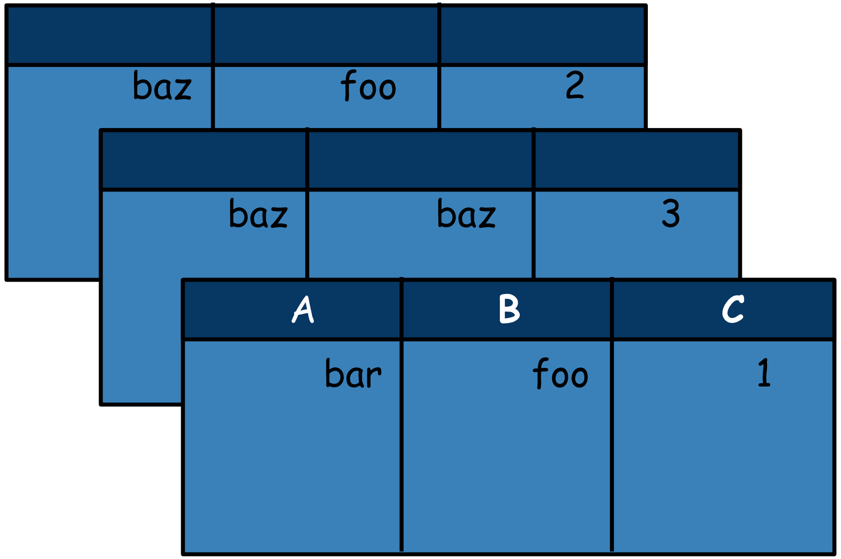


orderBy

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

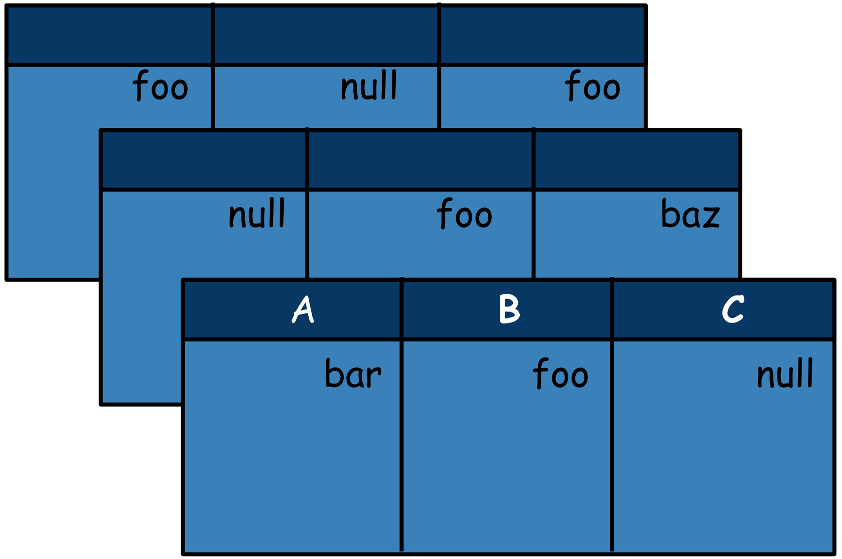


A	B	C
baz	foo	2
bar	foo	1
baz	baz	3

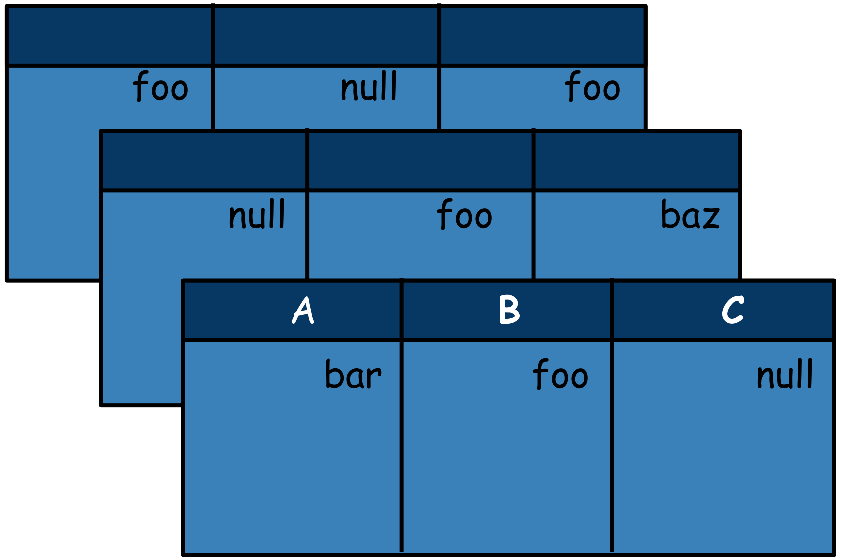


A	B	C
bar	foo	1
baz	baz	3
baz	foo	2

persist



printSchema

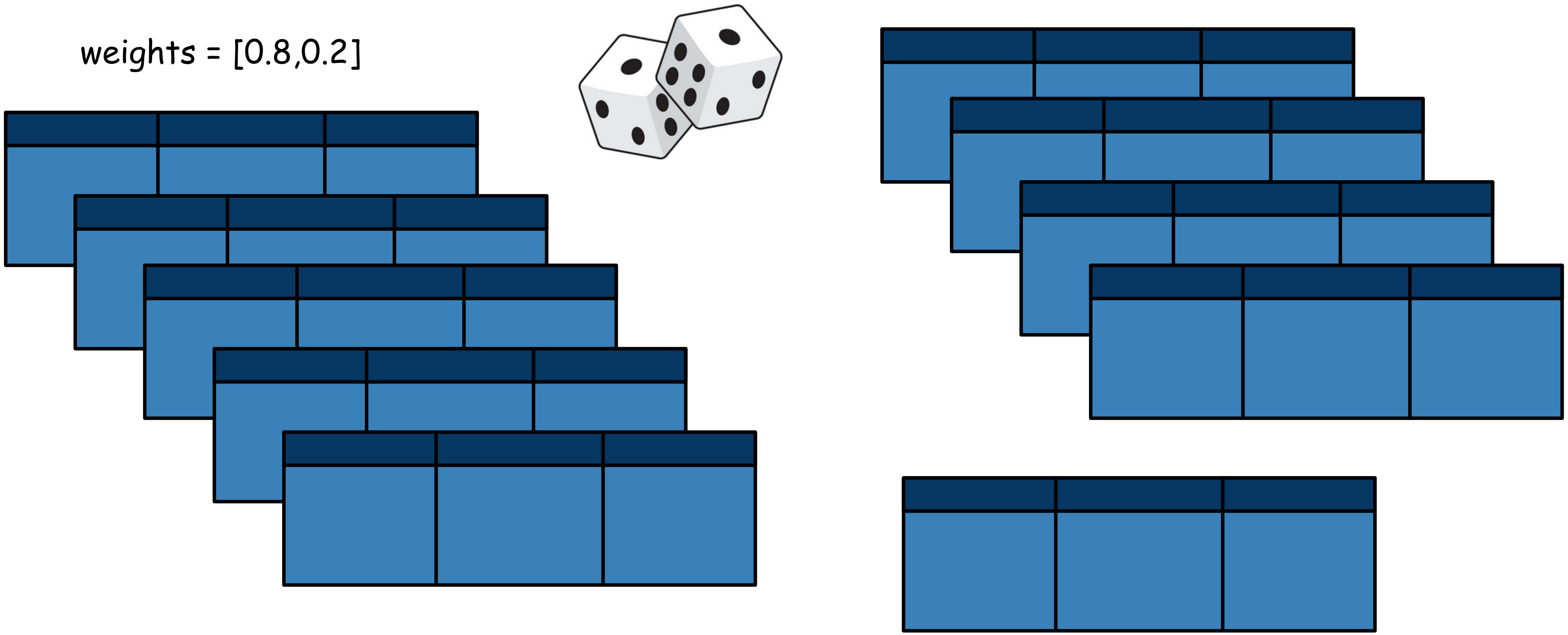


stdout

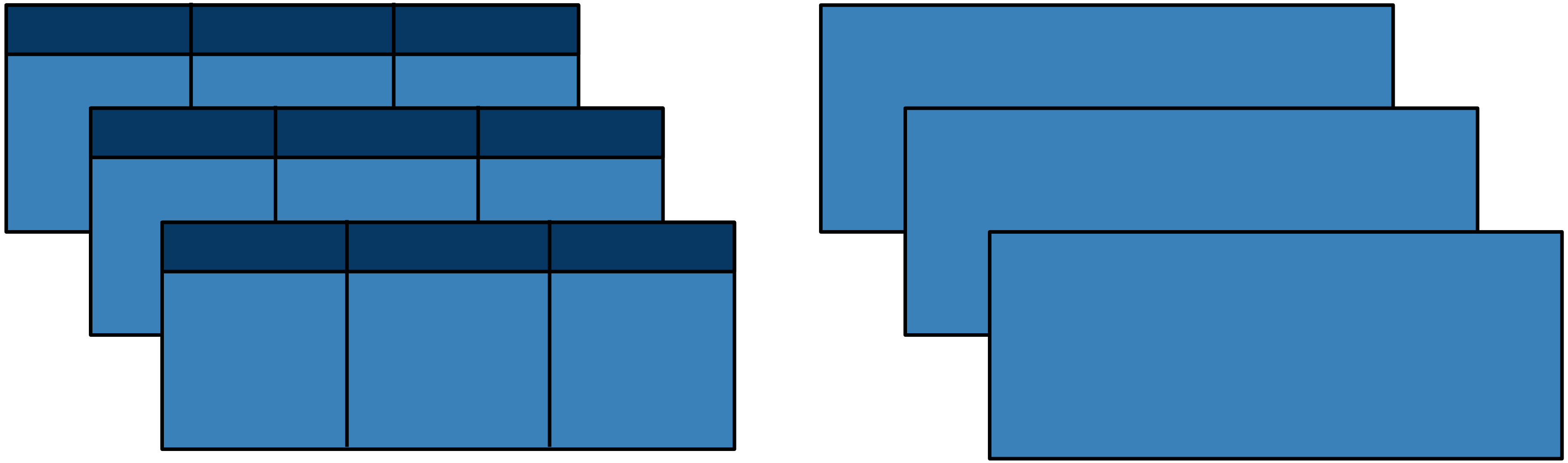
```
root
 |-- A: string (nullable = true)
 |-- B: string (nullable = true)
 |-- C: string (nullable = true)
```

randomSplit

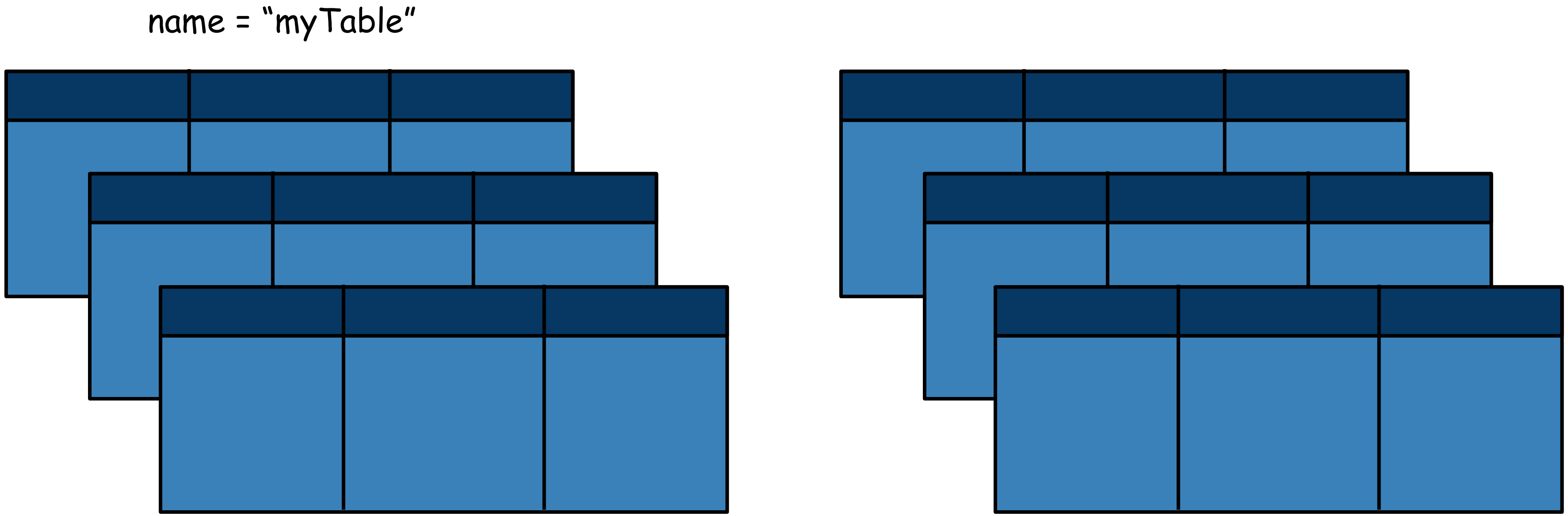
weights = [0.8,0.2]



rdd

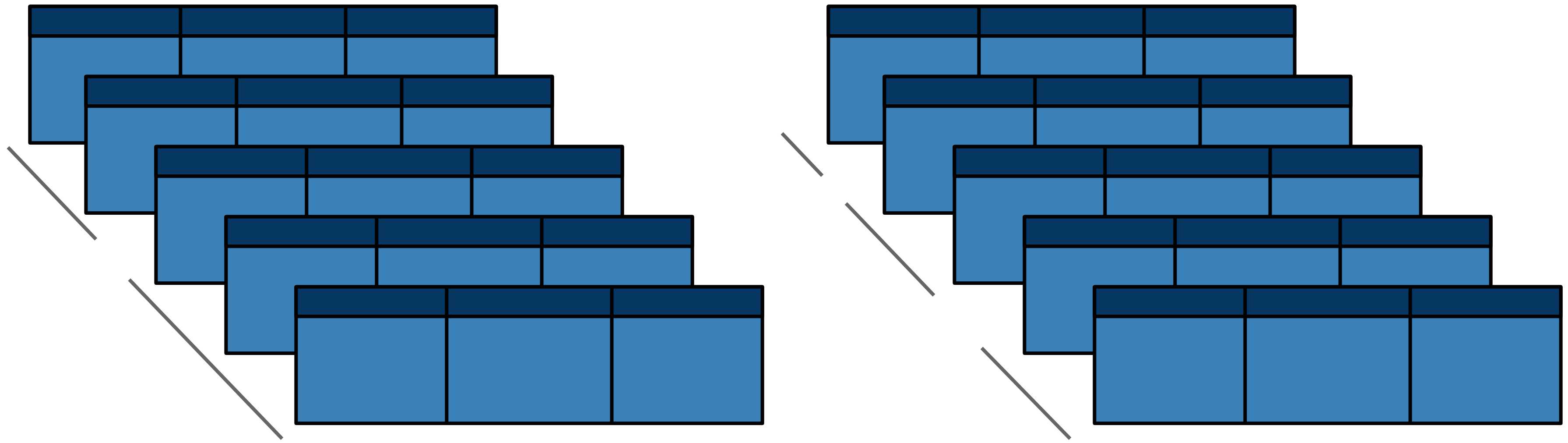


registerTempTable



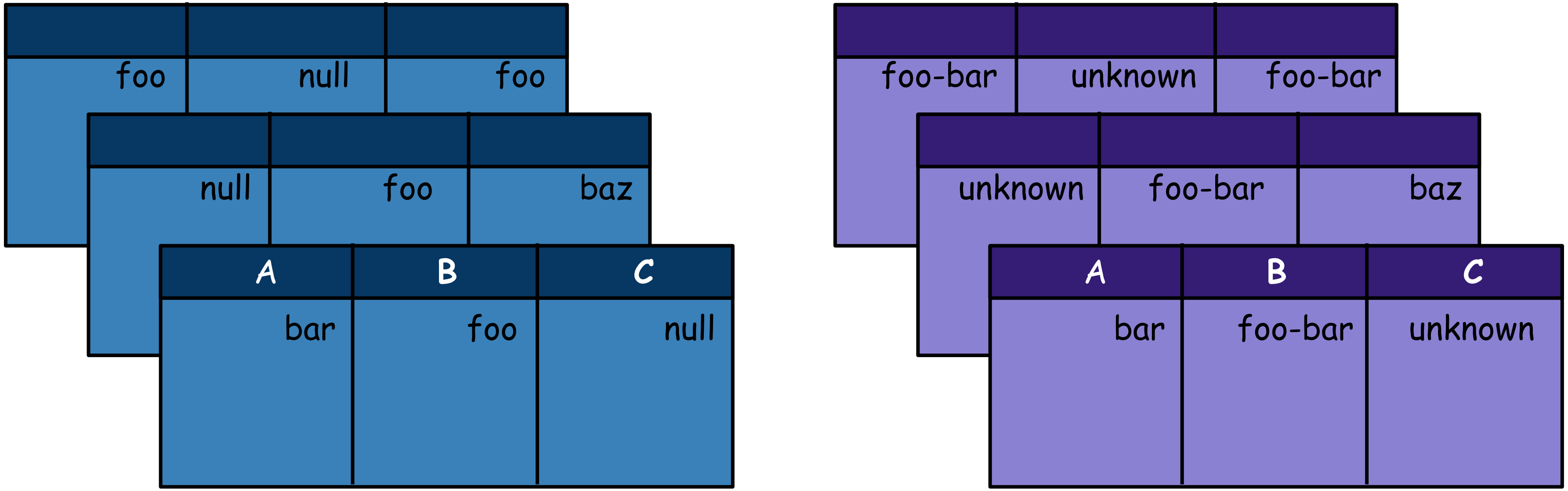
repartition

numPartitions = 3

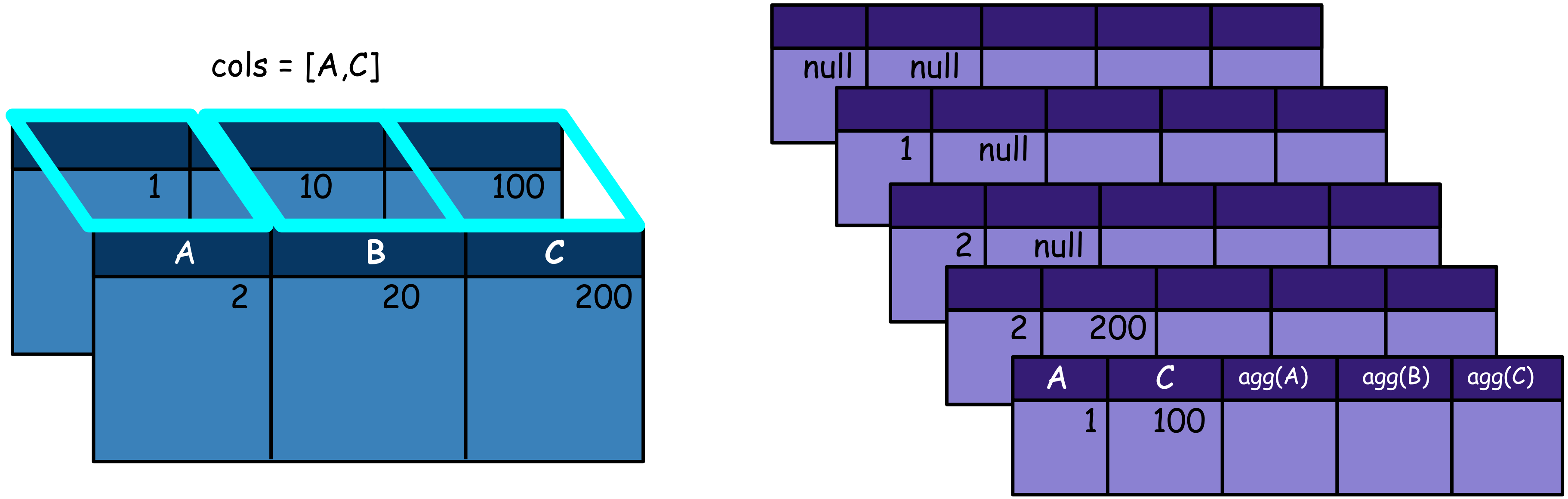


replace

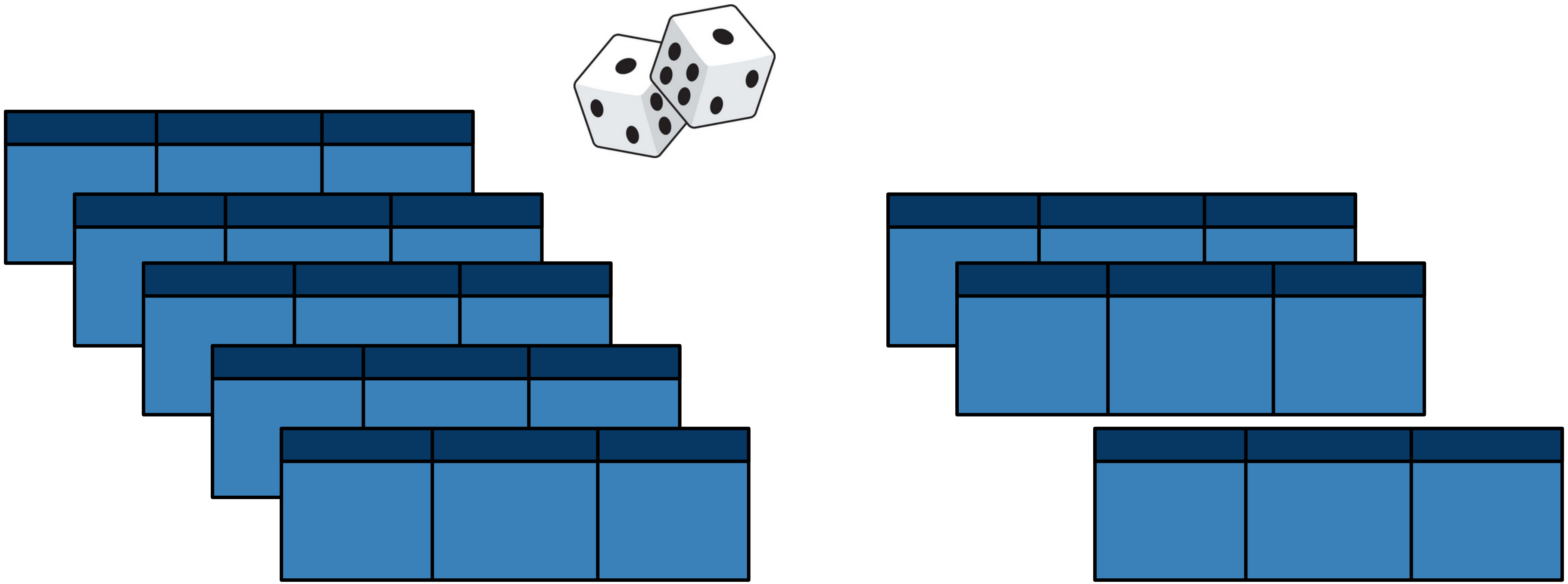
to_replace = ['foo', 'null'] value = ['foo-bar', 'unknown']



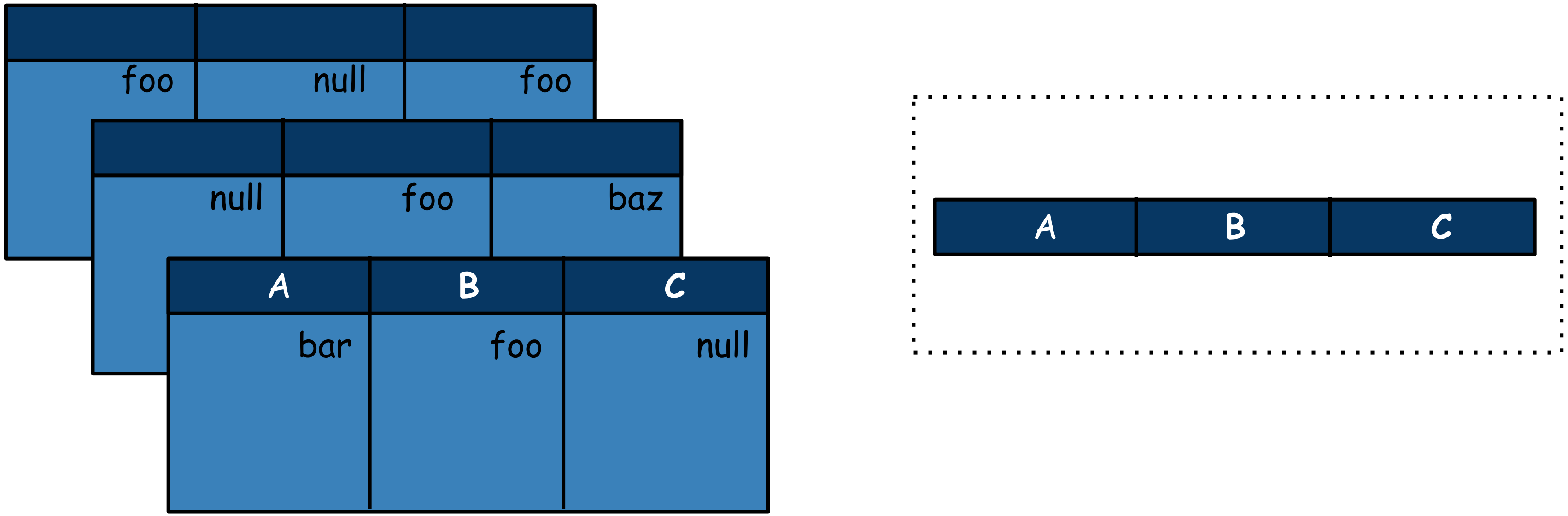
rollup


cols = [A,C]
1
10
100
A
B
C
2
20
200
null
null
1
null
2
null
2
200
A
C
agg(A)
agg(B)
agg(C)
1
100

sample

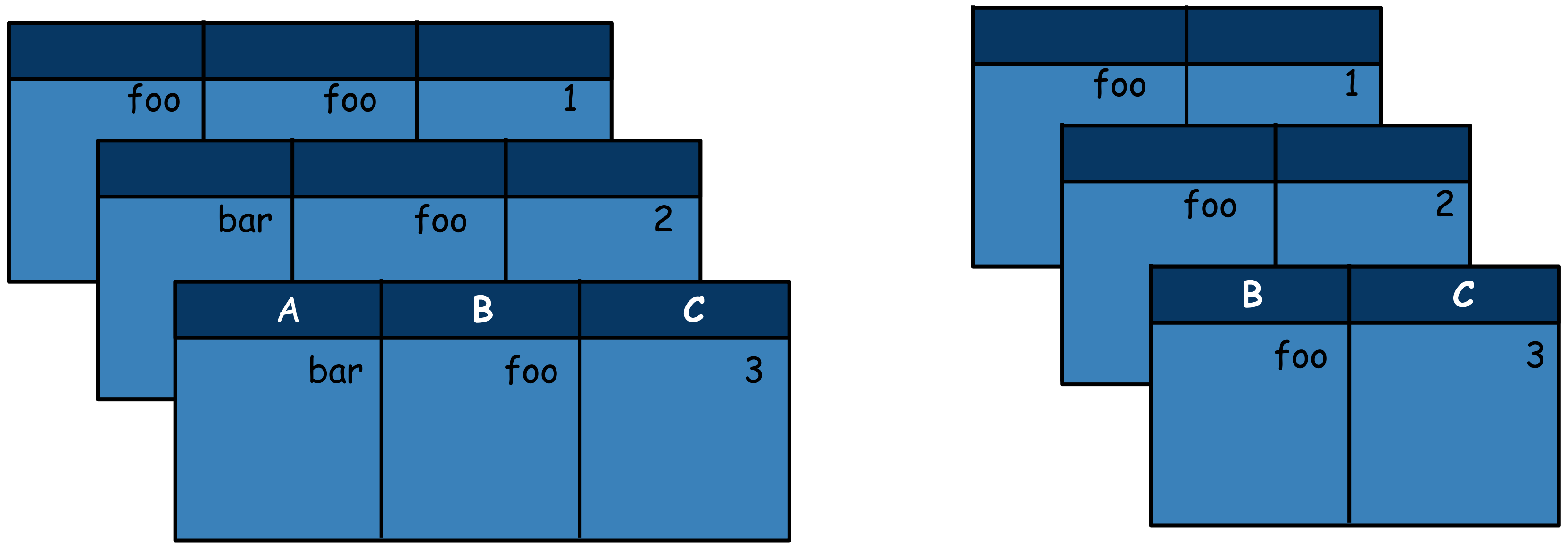


schema



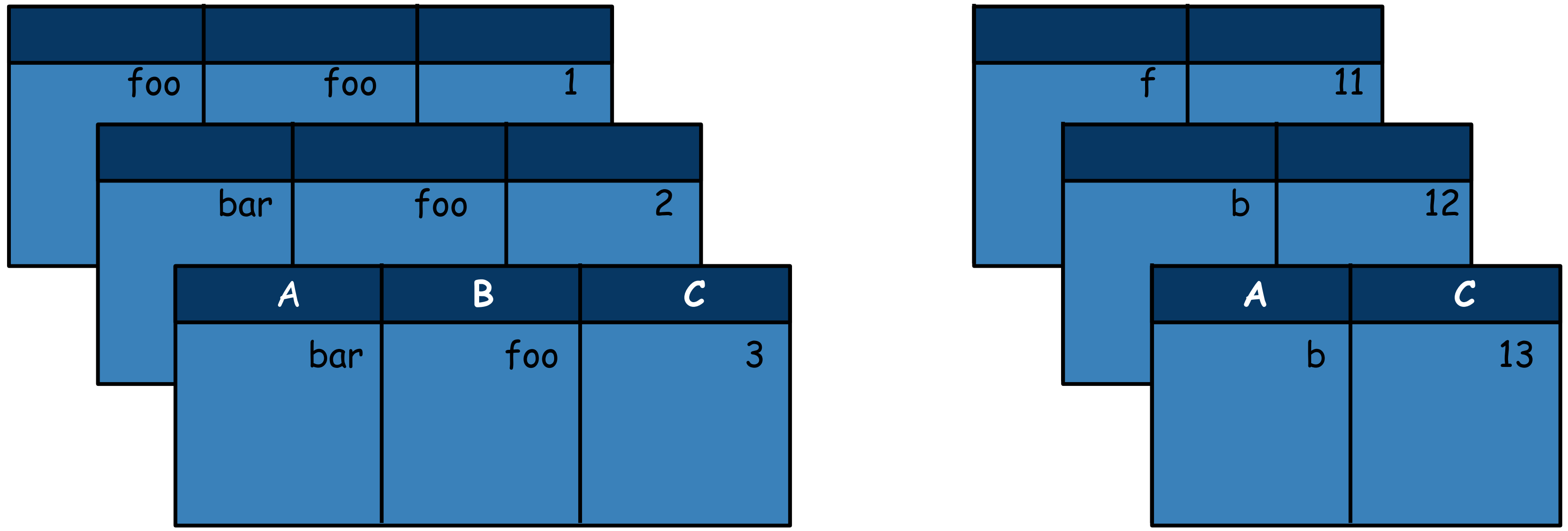
select

cols = ['B', 'C']

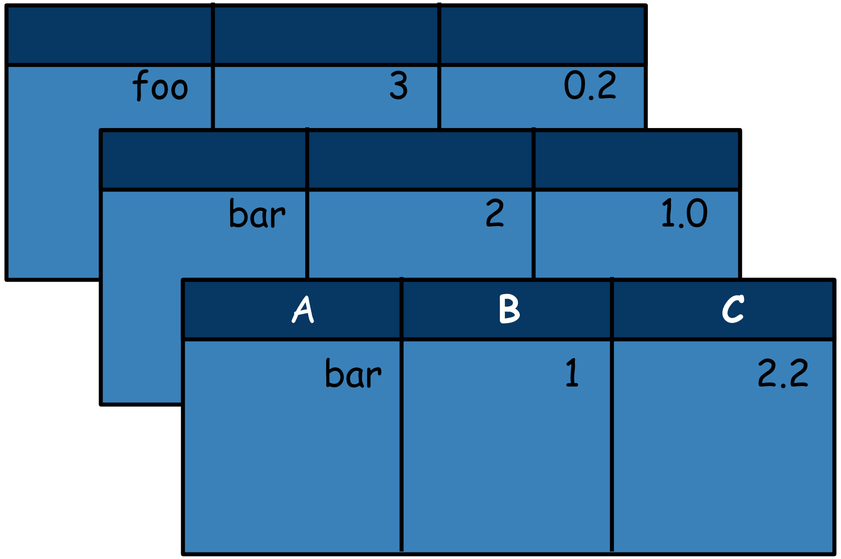


selectExpr

expr = [“substr(A,1,1)”, “C + 10”]



show

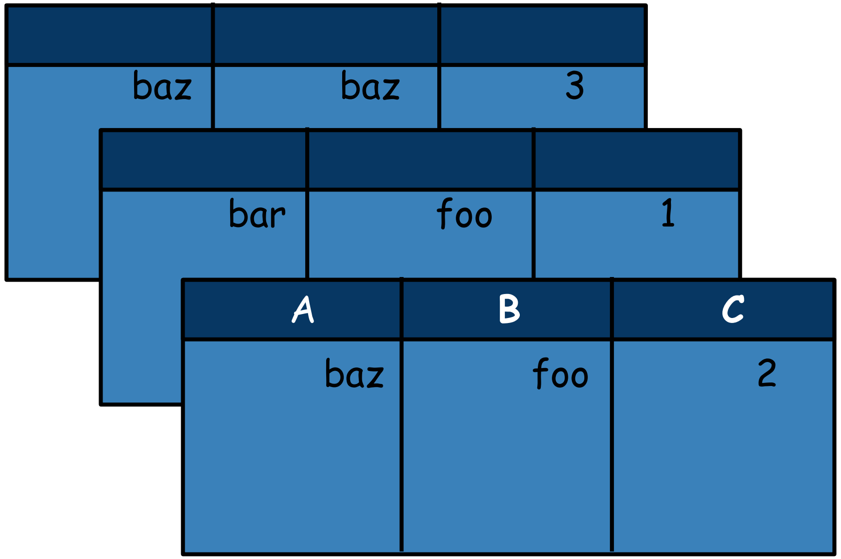


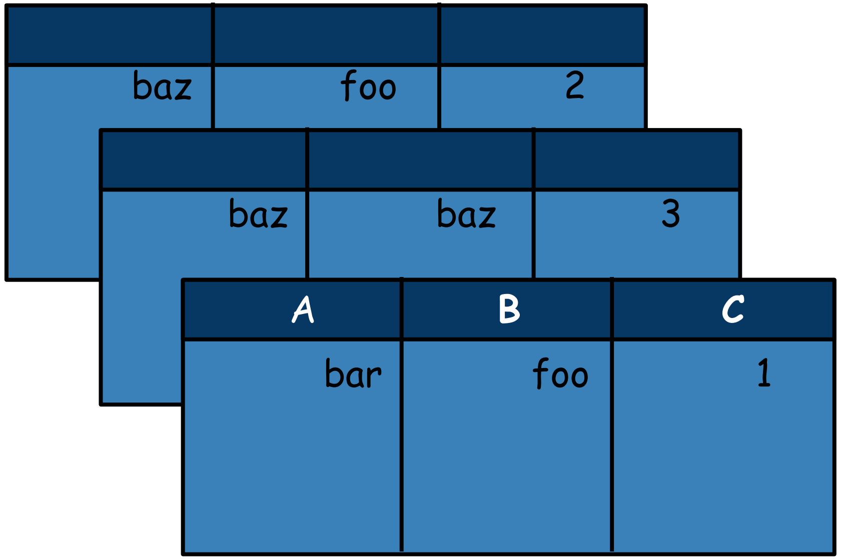
stdout

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

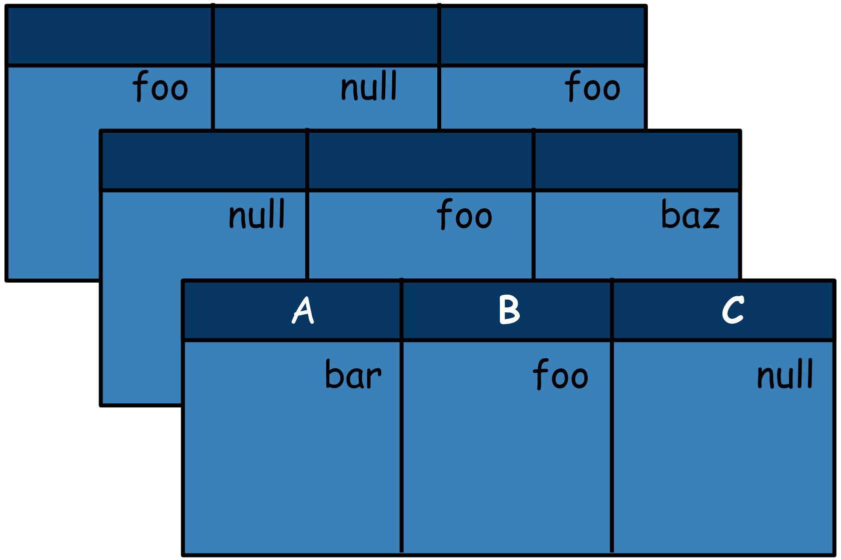
sort

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

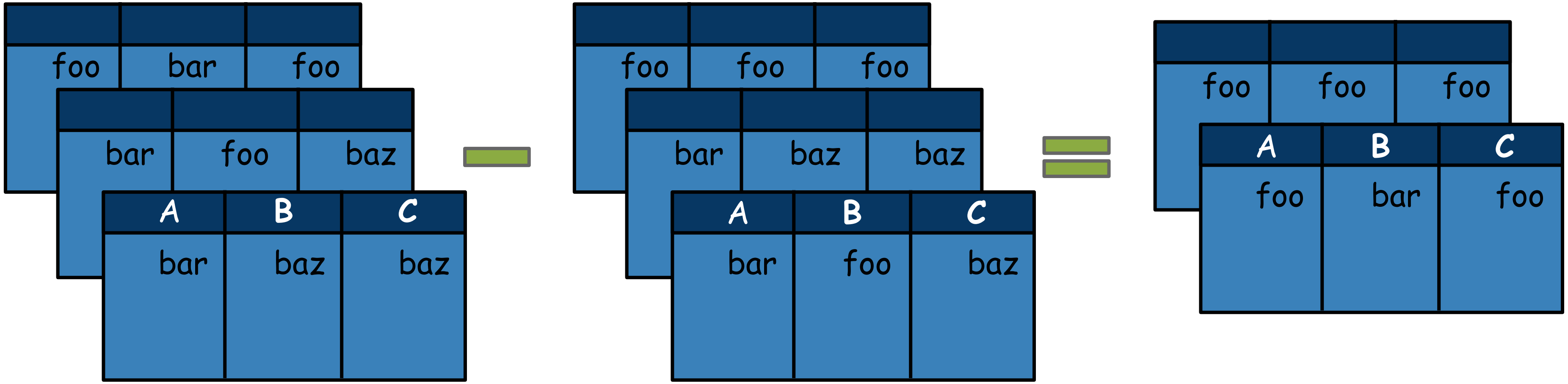




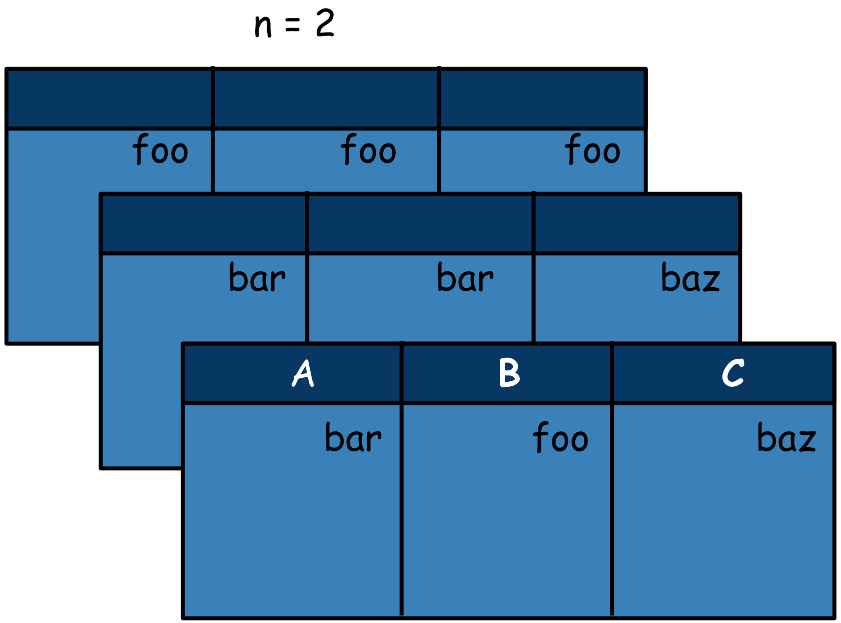
stat



subtract

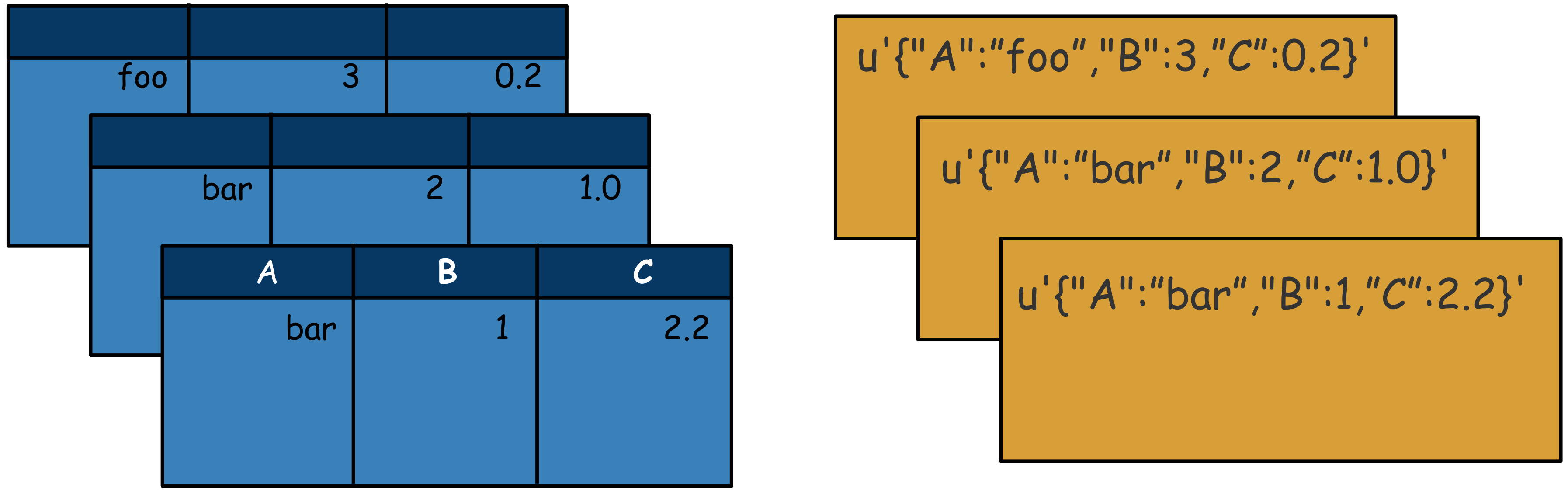

foo
bar
foo
bar
foo
baz
A
B
C
bar
baz
baz
foo
foo
foo
bar
baz
baz
A
B
C
bar
foo
baz
foo
foo
foo
A
B
C
foo
bar
foo

take



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

toJSON



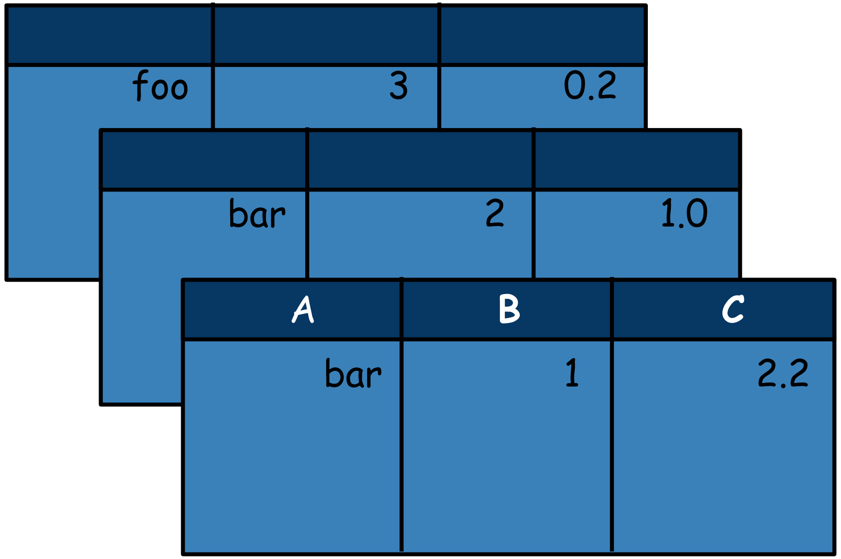
foo	3	0.2

bar	2	1.0

A	B	C
bar	1	2.2

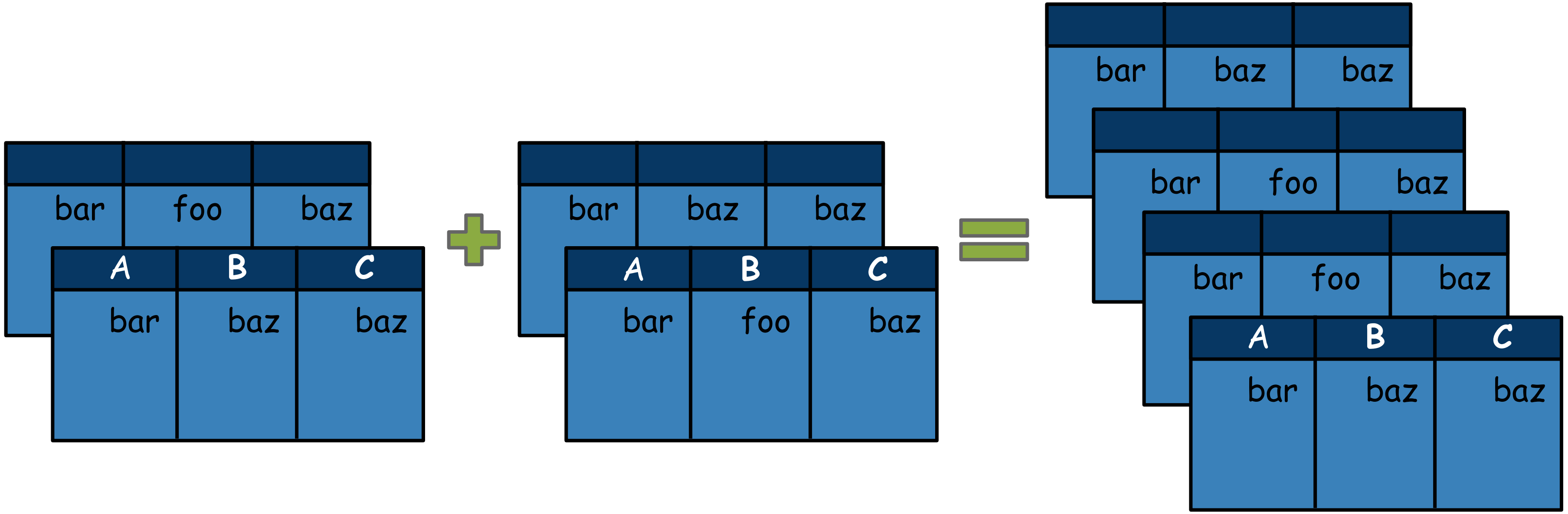
```
u'{"A":"foo","B":3,"C":0.2}'
u'{"A":"bar","B":2,"C":1.0}'
u'{"A":"bar","B":1,"C":2.2}'
```

toPandas



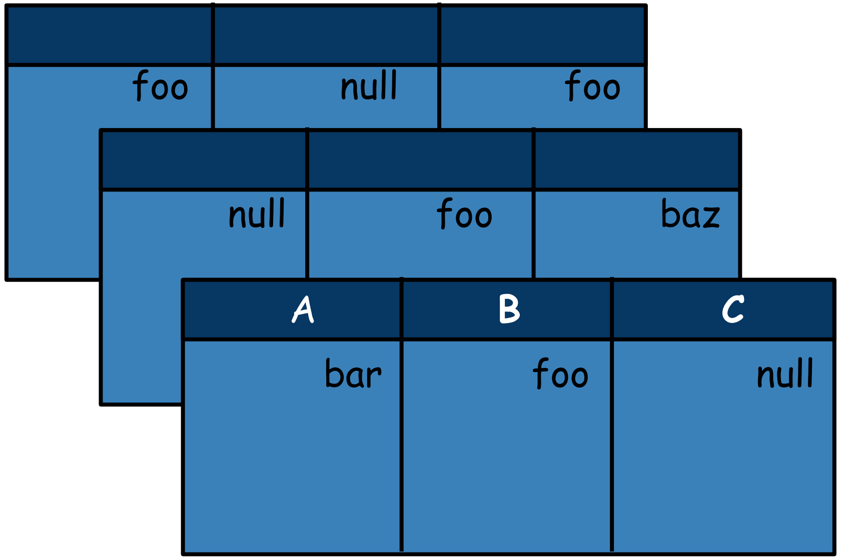
```
    A   B   C
0  foo  3  0.2
1  bar  2  1.0
2  bar  1  2.2
```

unionAll

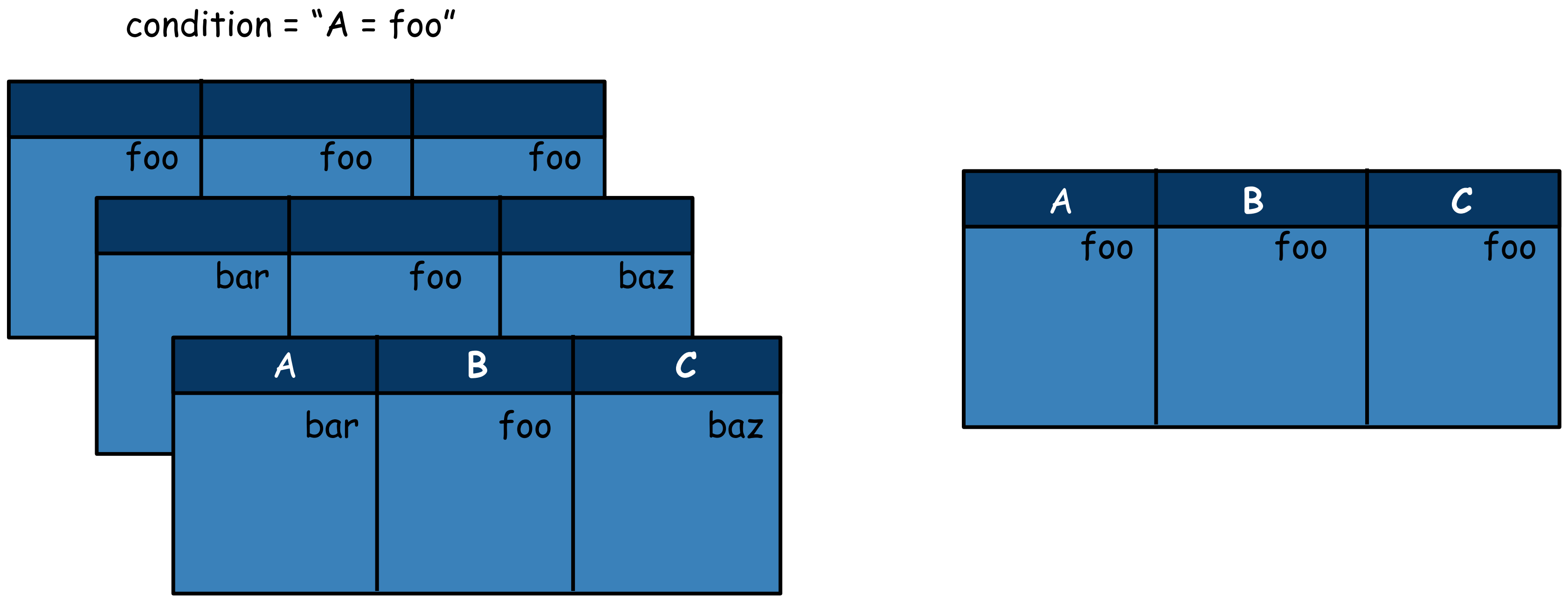


bar
foo
baz
A
B
C
bar
baz
baz
bar
baz
baz
A
B
C
bar
foo
baz
bar
baz
baz
bar
foo
baz
bar
foo
baz
A
B
C
bar
baz
baz

unpersist

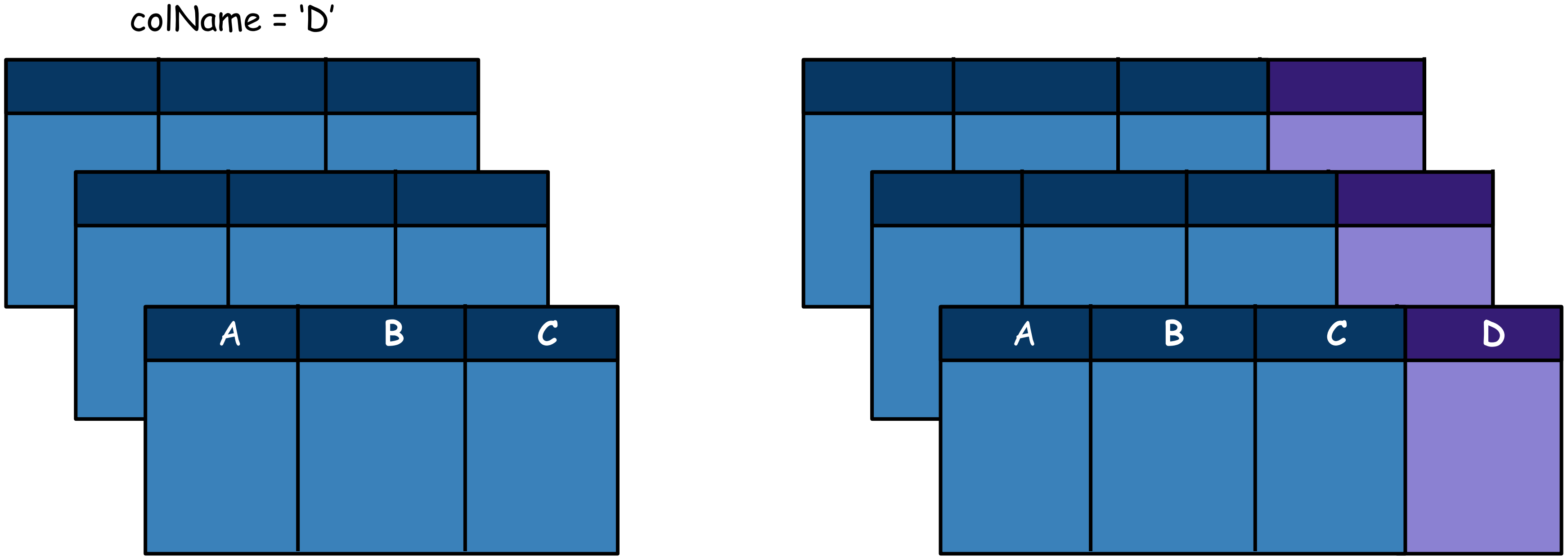


where



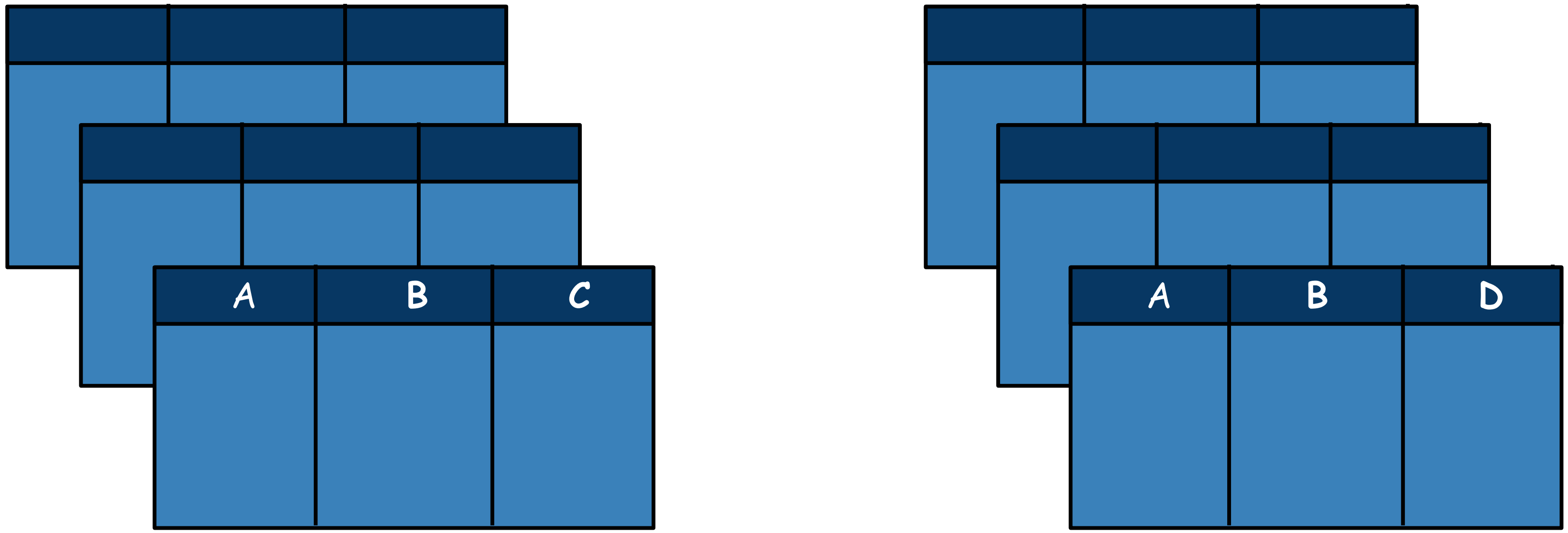
condition = “A = foo”
foo
foo
foo
bar
foo
baz
A
B
C
bar
foo
baz
A
B
C
foo
foo
foo

withColumn

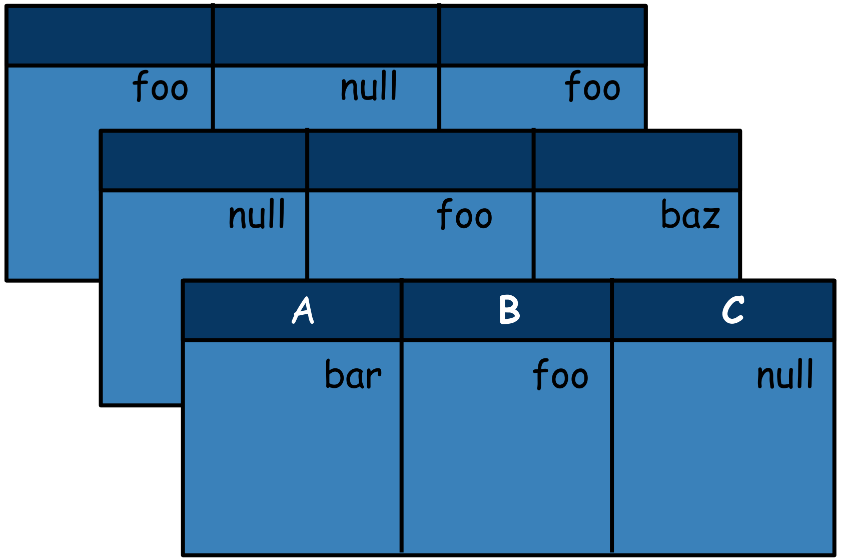


withColumnRenamed

existing = 'C' col = 'D'



write


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