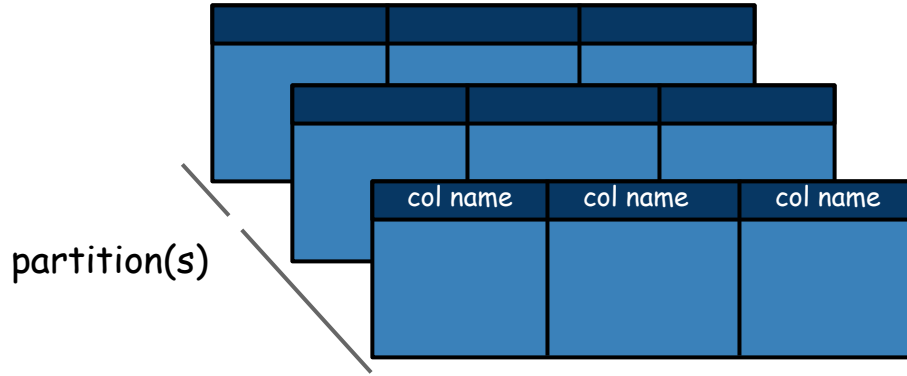


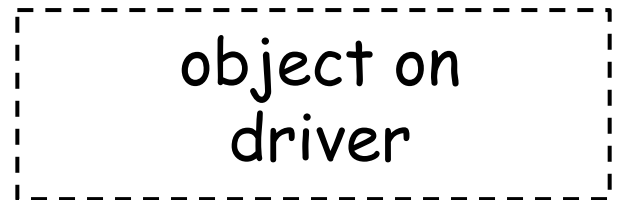
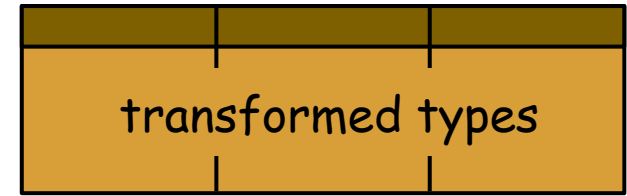
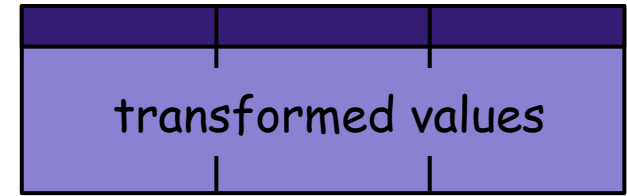
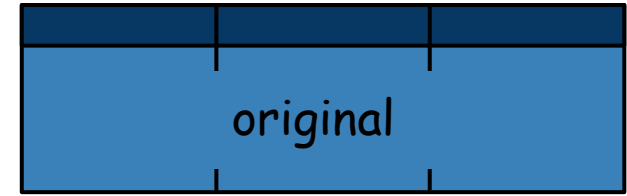
# pyspark-pictures data frames

Learn the pyspark API through pictures and simple examples

data frame



row



user input

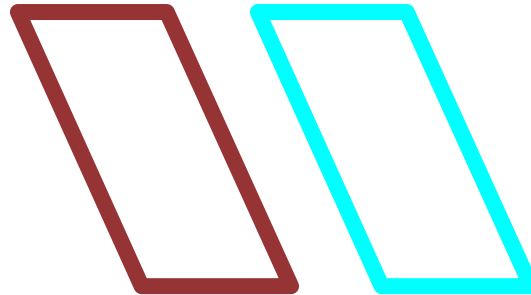
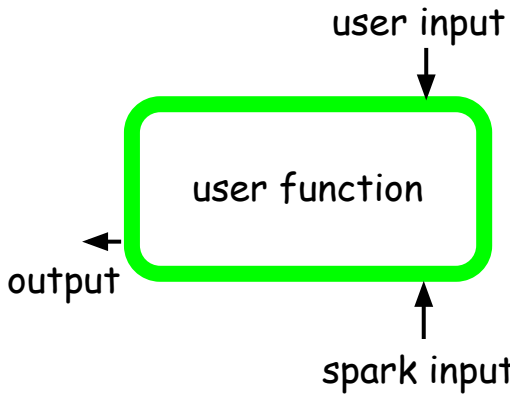
groupby  
function

aggregate  
function

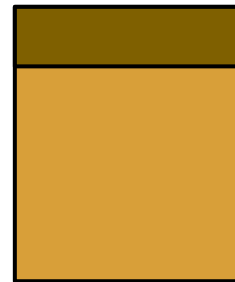
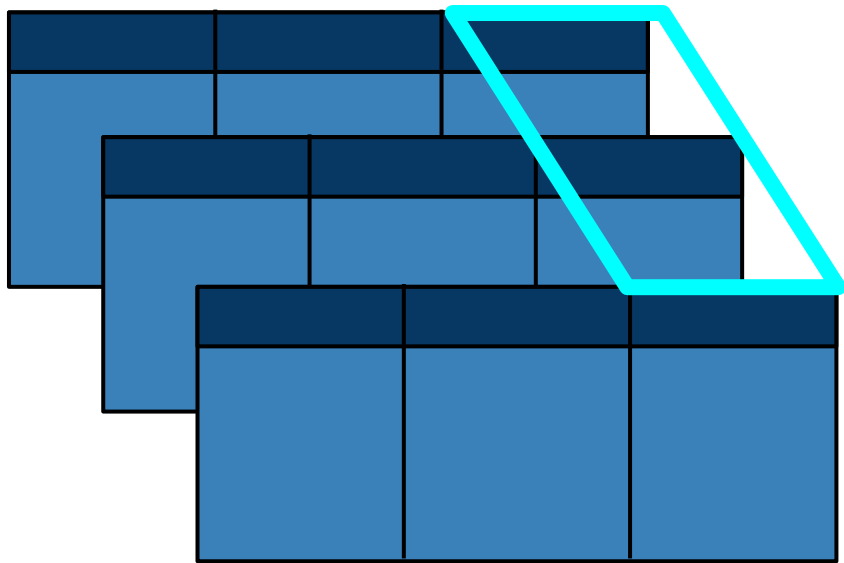
user function

spark input

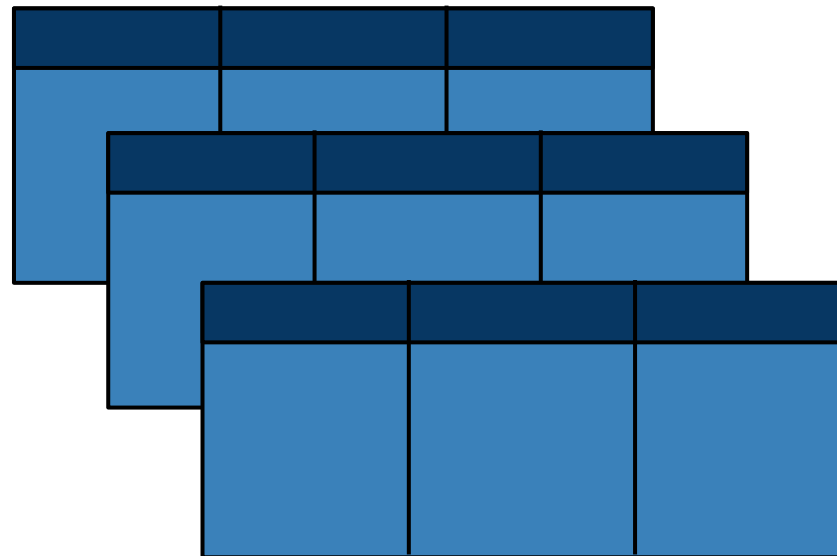
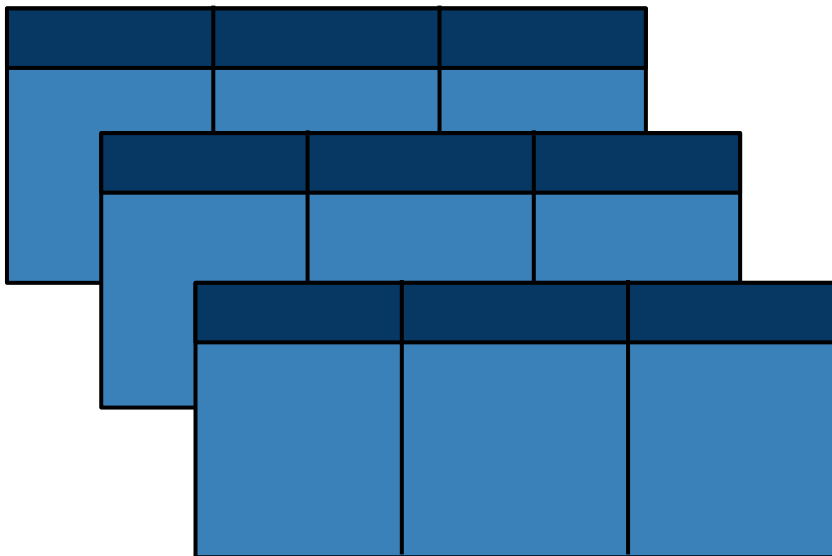
output



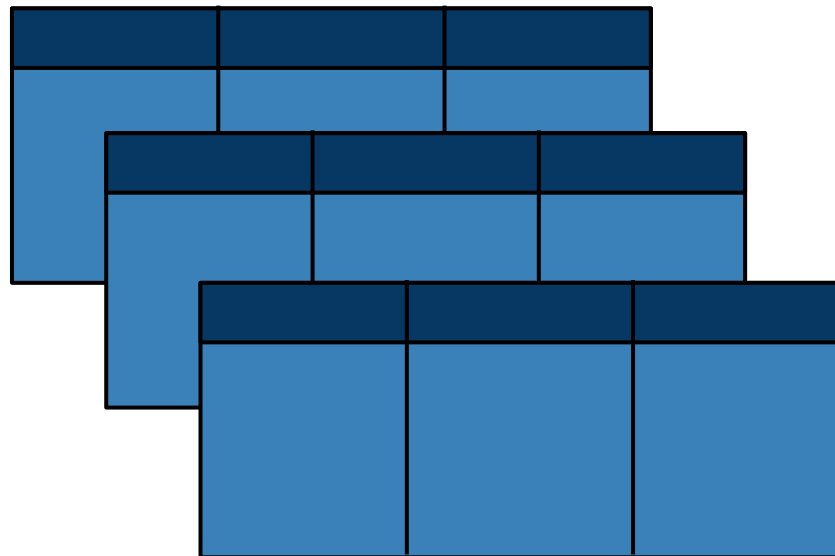
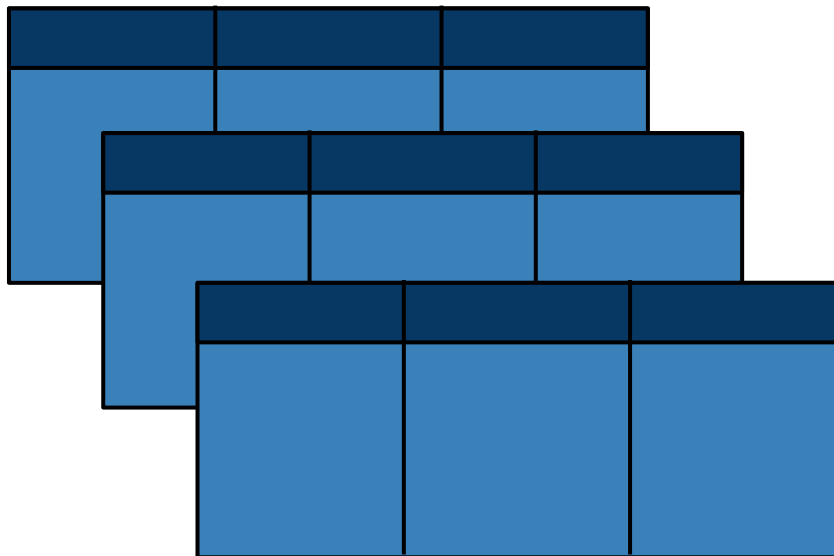
agg



# alias

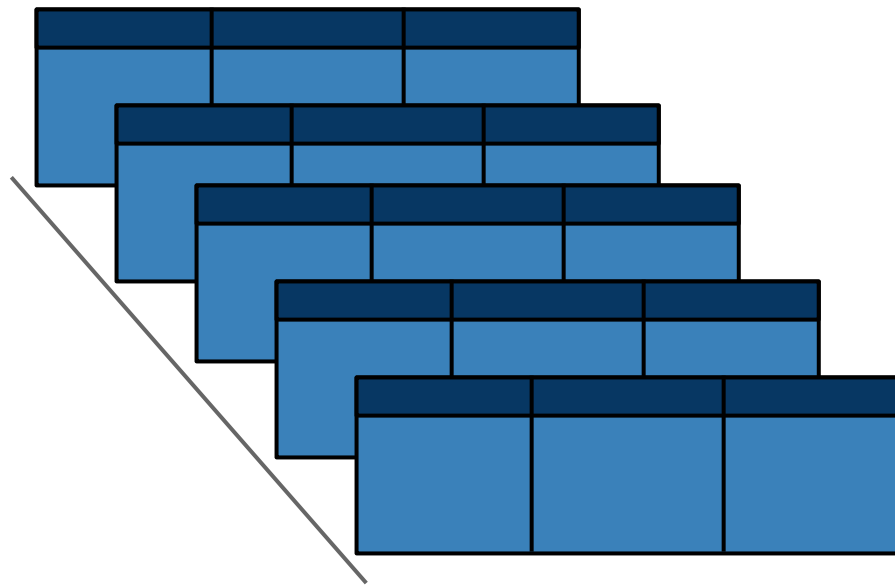
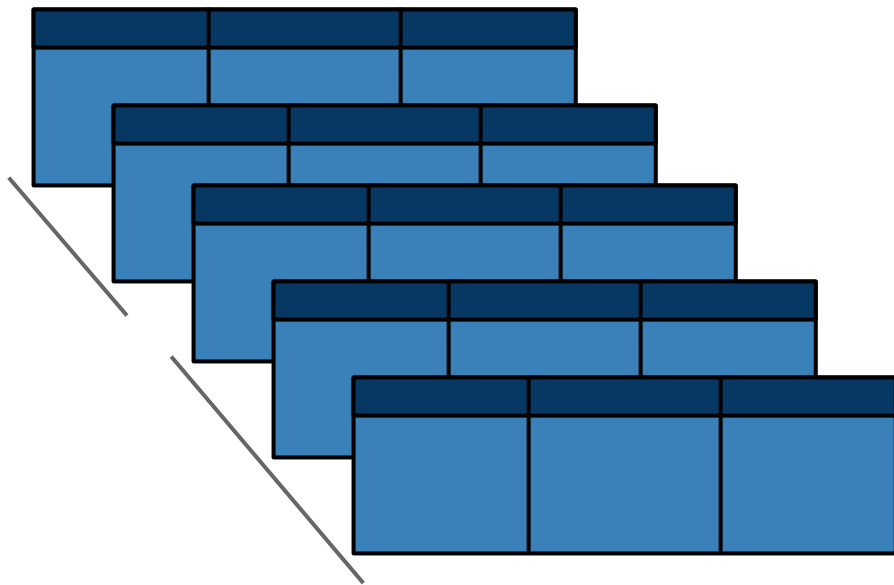


# cache

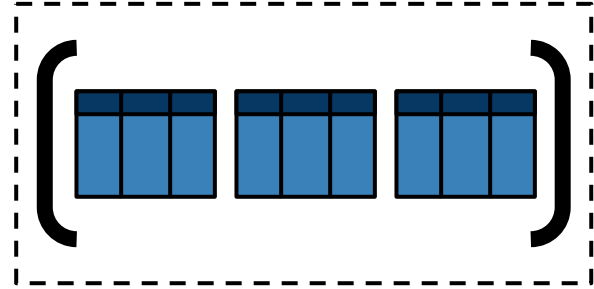
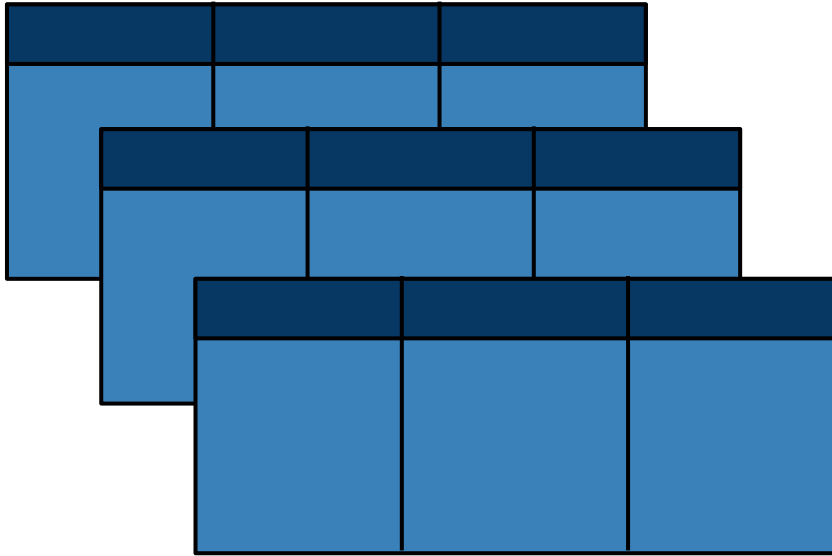


# coalesce

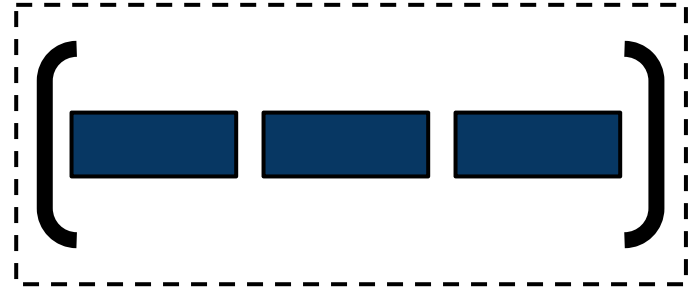
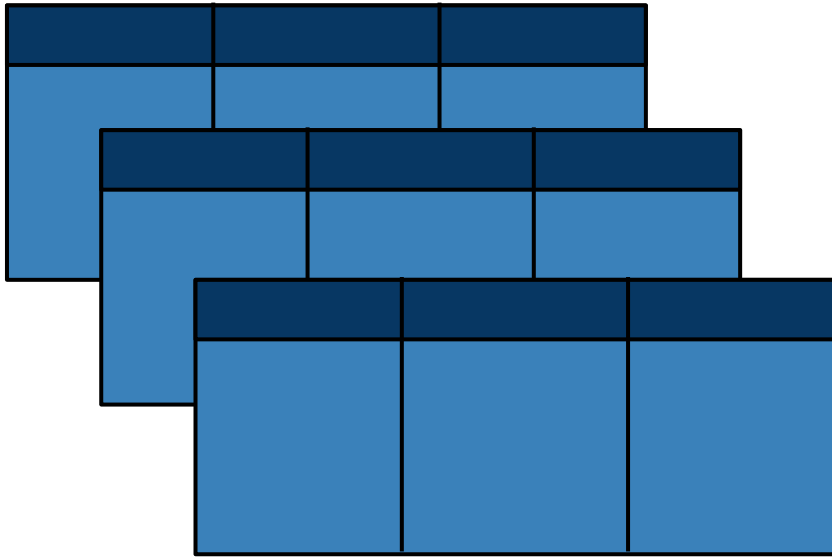
numPartitions = 1



# collect

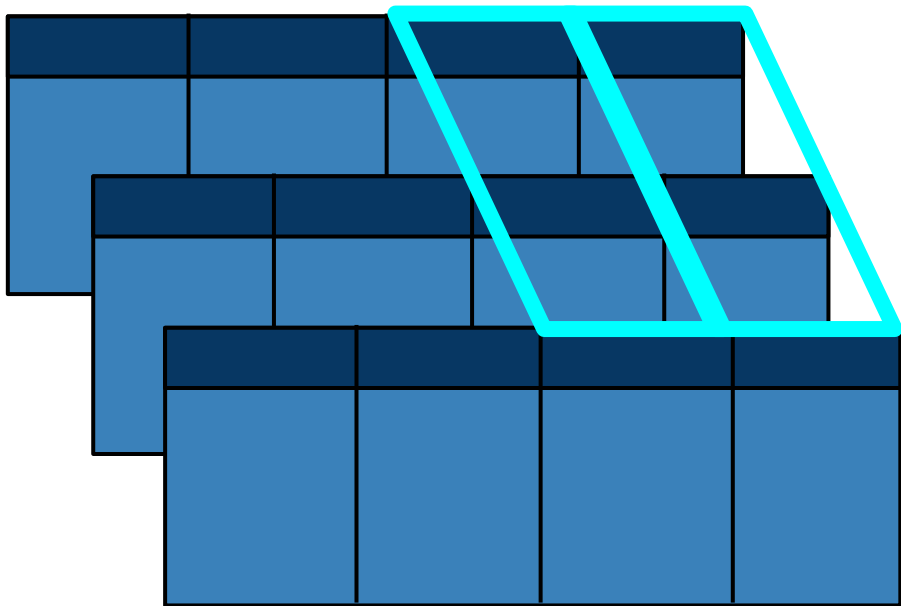


# columns





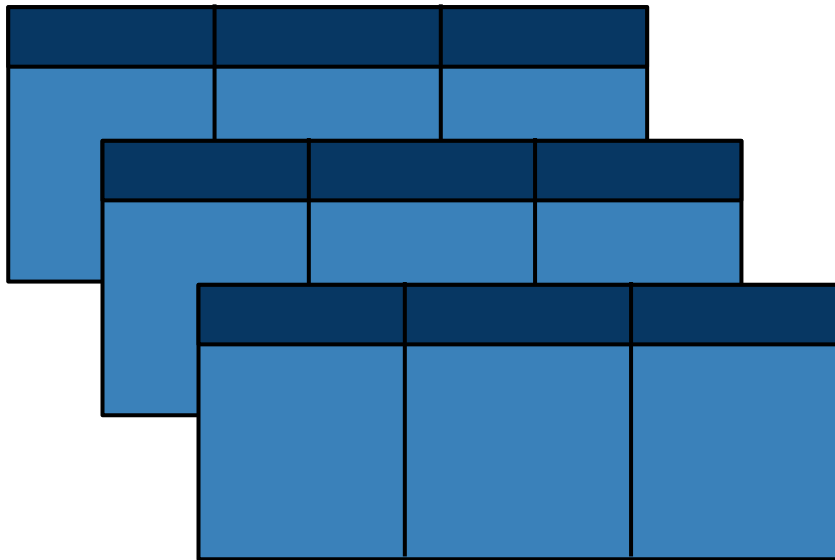
# corr



Pearson's r

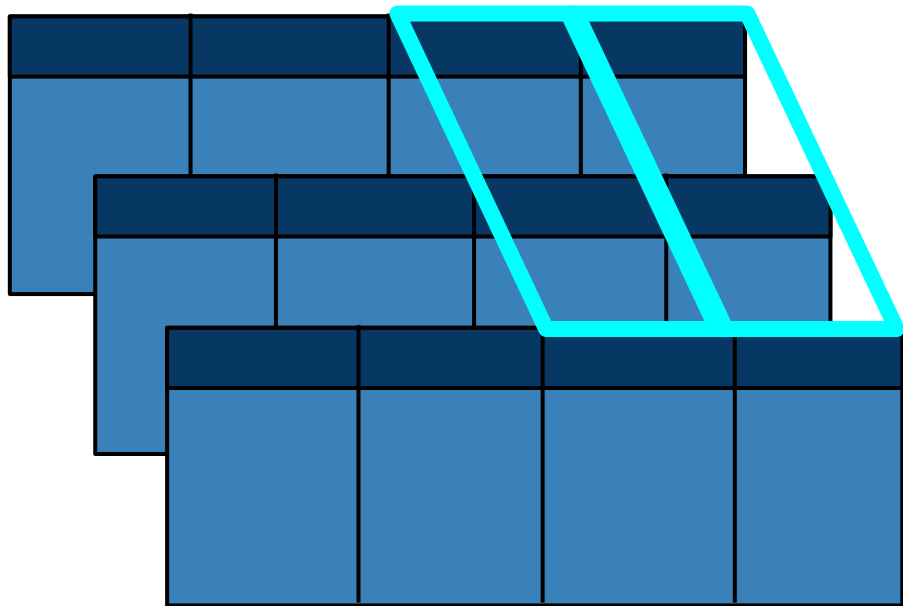
$$r = \frac{\sum_i (A_i - \bar{A})(C_i - \bar{C})}{\sqrt{\sum_i (A_i - \bar{A})^2} \sqrt{\sum_i (C_i - \bar{C})^2}}$$

# count



3

# COV

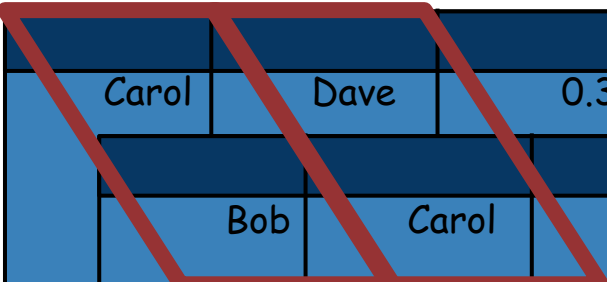


Sample Covariance

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

# crosstab

col1 = 'from' col2 = 'to'



Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	0	0	1
Bob	0	1	0
from_to	Bob	Carol	Dave
Alice	1	0	0

# cube

\*cols = 'from', 'to'

Alice	Carol	0.2
from	to	amt
Alice	Bob	0.1

null	null	
Alice	null	
null	Carol	
null	Bob	
Alice	Carol	
from	to	agg(amt)
Alice	Bob	

# describe

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

max	
min	
stdev	
mean	
summary	amt
count	

# distinct

Bob	Carol	0.2
Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

# drop

col = 'amt'

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	Dave
Bob	Carol
from	to
Alice	Bob



# dropDuplicates

subset = ['from', 'to']

Bob	Carol	0.2
Bob	Carol	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

# dropna

how = 'any' subset = ['from', 'to']

Bob	Carol	0.2
Carol	null	0.3
Bob	Carol	null
from	to	amt
null	Bob	0.1

Bob	Carol	0.2
from	to	amt
Bob	Carol	null

# dtypes

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

[('from','string'), ('to','string'), ('amt','double')]

# explain

extended = True

Carol	Dave	0.3

Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

== Parsed Logical Plan ==

...

== Analyzed Logical Plan ==

...

== Optimized Logical Plan ==

...

== Physical Plan ==

...

== RDD ==

# fillna

value = 'unknown' subset = ['from', 'to']

Carol	null	0.3

Bob	Carol	nan

from	to	amt
null	Bob	0.1

Carol	unknown	0.3

Bob	Carol	nan

from	to	amt
unknown	Bob	0.1

# filter

condition = "amt > 0.1"

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

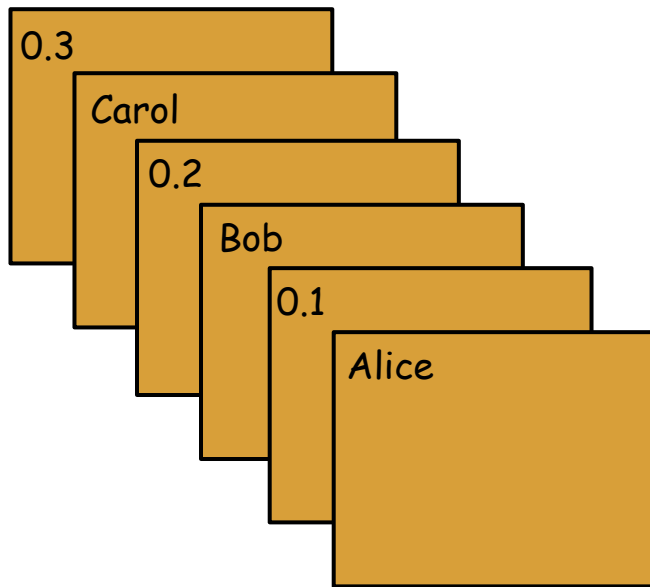
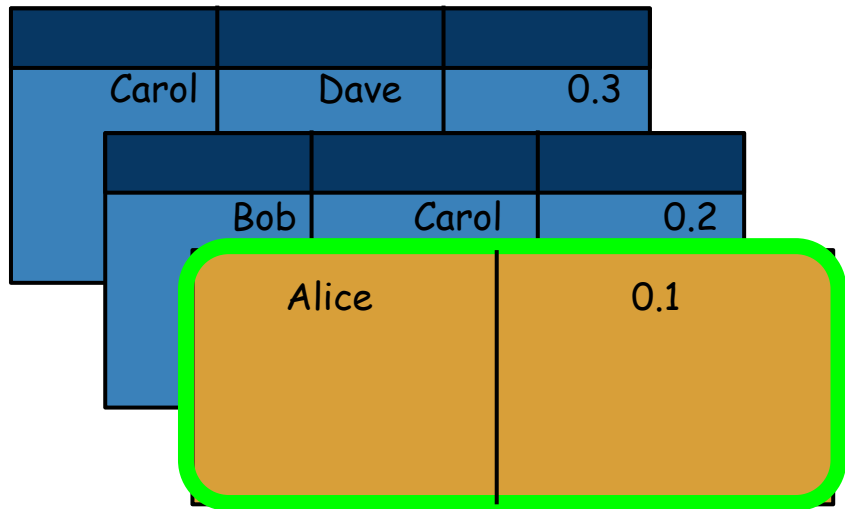
Carol	Dave	0.3
from	to	amt
Bob	Carol	0.2

# first

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Row(from='Alice', to='Bob', amt=0.1)

# flatMap





# foreach

Carol	Dave	0.3

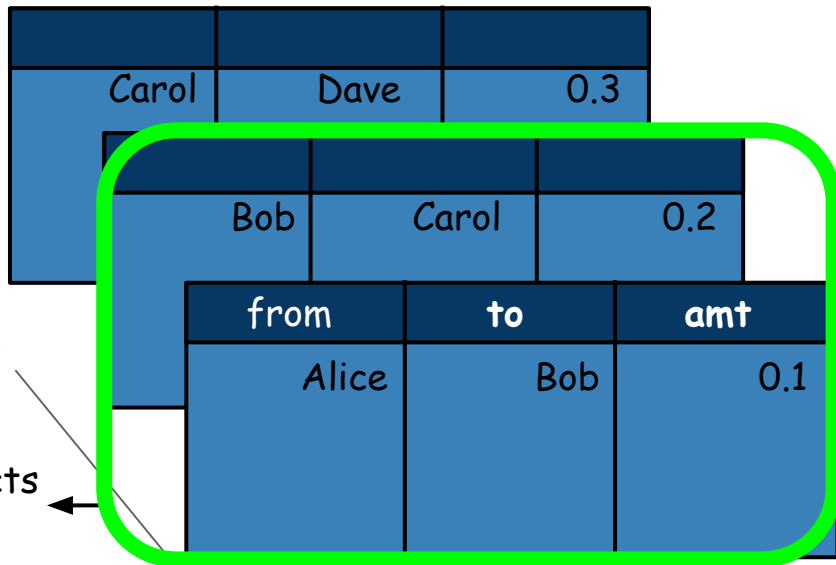
Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

side effects  
(e.g print) ←

\*no return value,  
original DataFrame  
unchanged

# foreachPartition



The diagram illustrates the `foreachPartition` operation. It shows three nested DataFrames. The outermost DataFrame has a header row and one data row (Carol, Dave, 0.3). The middle DataFrame also has a header row and one data row (Bob, Carol, 0.2). The innermost DataFrame has a header row and one data row (Alice, Bob, 0.1). A green rounded rectangle highlights the innermost DataFrame, indicating it is the current partition being processed. An arrow points from the text 'side effects (e.g print)' to the green box.

Carol	Dave	0.3

Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

side effects  
(e.g print)

\*no return value,  
original DataFrame  
unchanged

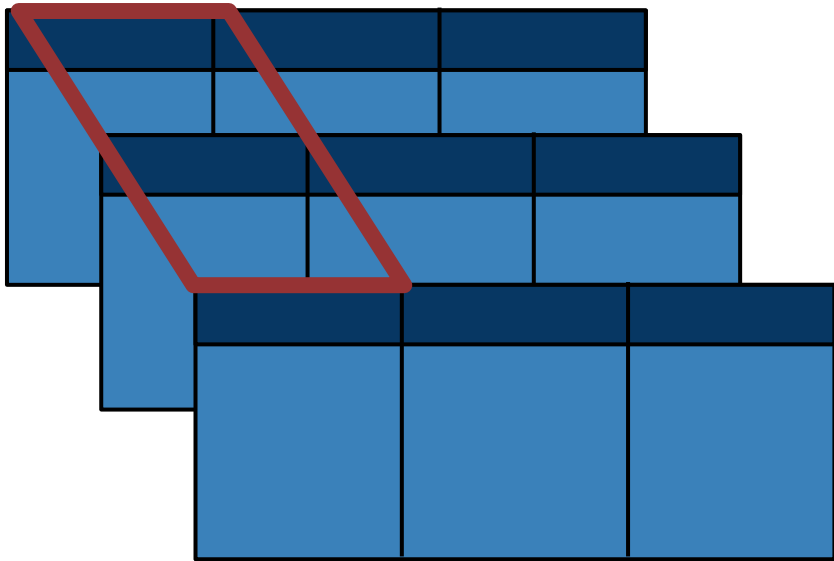
# freqItems

cols = ['from', 'amt'] support = 0.8

Carol	Bob	0.1
Alice	Bob	0.5
Alice	Bob	0.1
Alice	Dave	0.1
from	to	amt
Bob	Carol	0.1

from_freqItems	amt_freqItems
[Alice]	[0.1]

# groupBy (groupby)



*GroupedData Object*  
with methods: `agg`, `avg`, `count`,  
`max`, `mean`, `min`, `pivot`, `sum`

# groupBy(col1).avg(col2)

col1 = 'from' col2 = 'amt'

Carol	Dave	0.3
Alice	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	0.3
from	avg(amt)
Alice	0.15

# head

n = 2

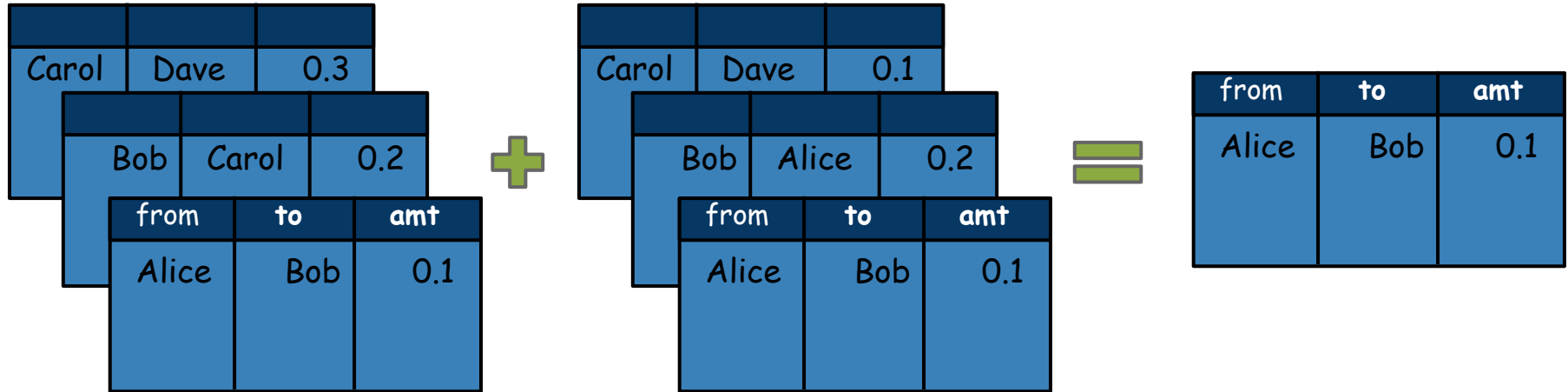
Carol	Dave	0.3

Bob	Carol	0.2

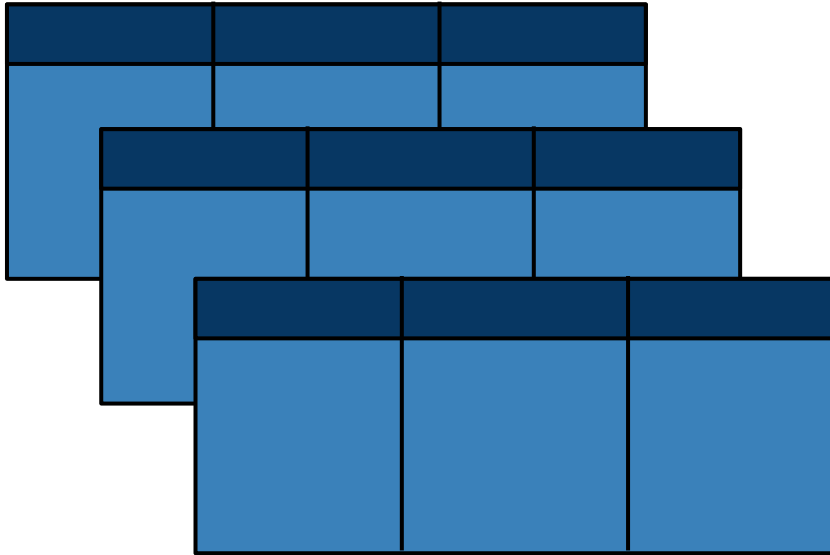
from	to	amt
Alice	Bob	0.1

[Row(from=u'Alice', to=u'Bob', amt=0.1),  
Row(from=u'Bob', to=u'Carol', amt=0.2)]

# intersect



# isLocal

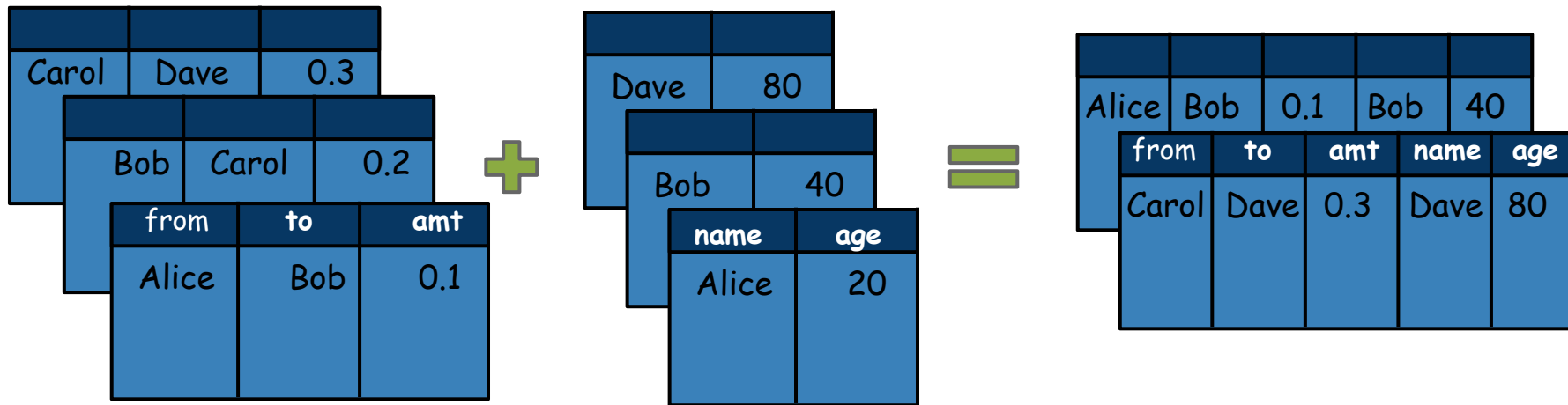


False



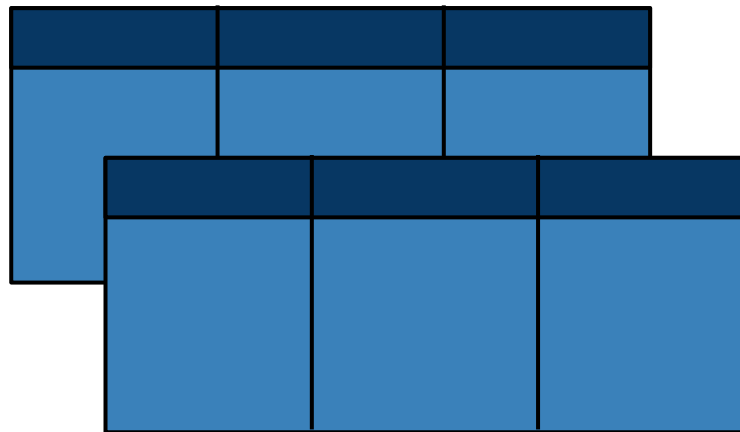
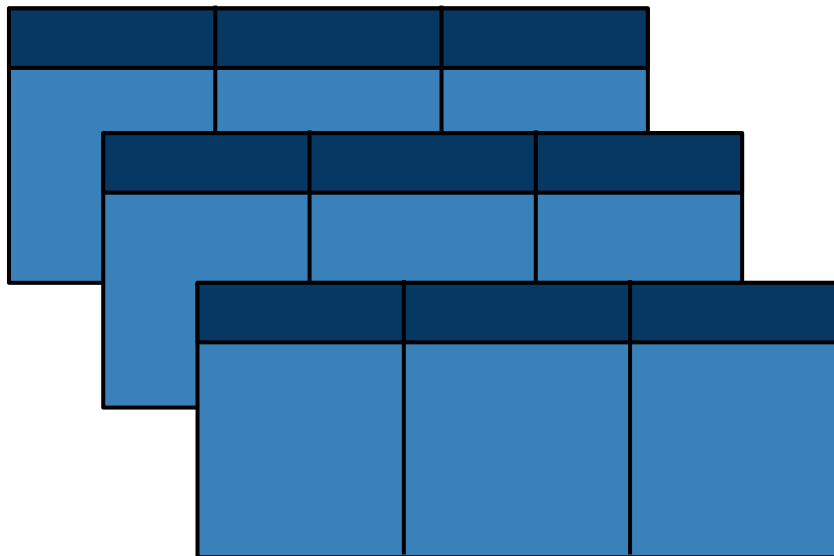
# join

joinExprs = x.to==y.name joinType = 'inner'

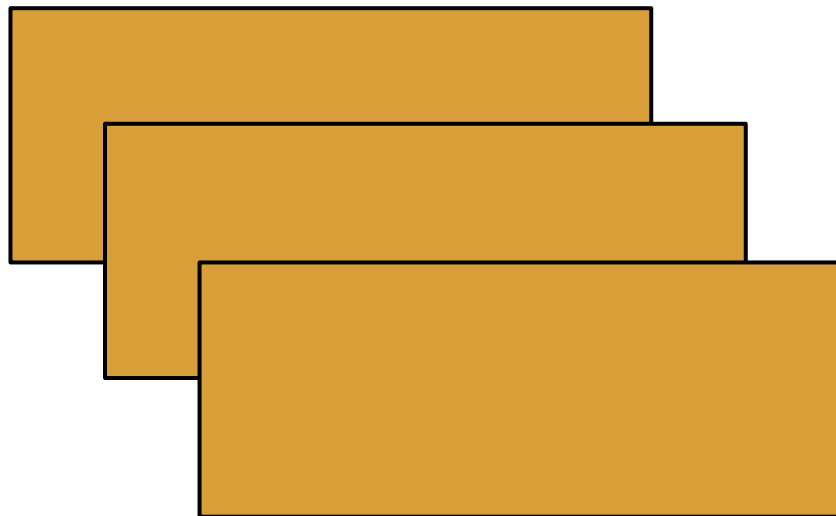


# limit

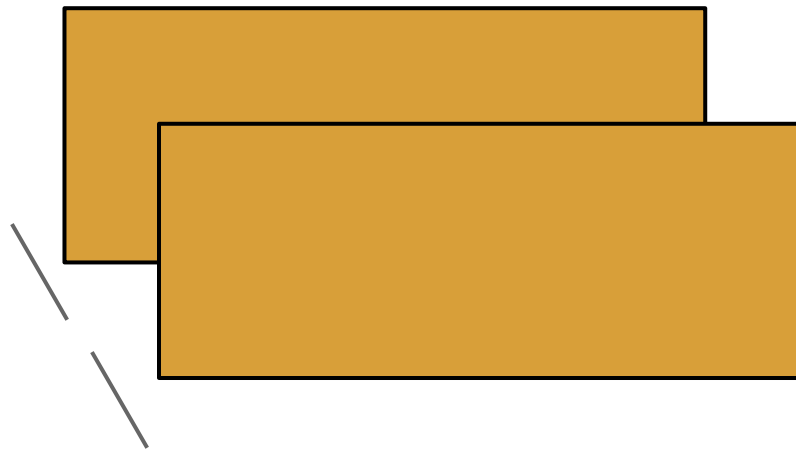
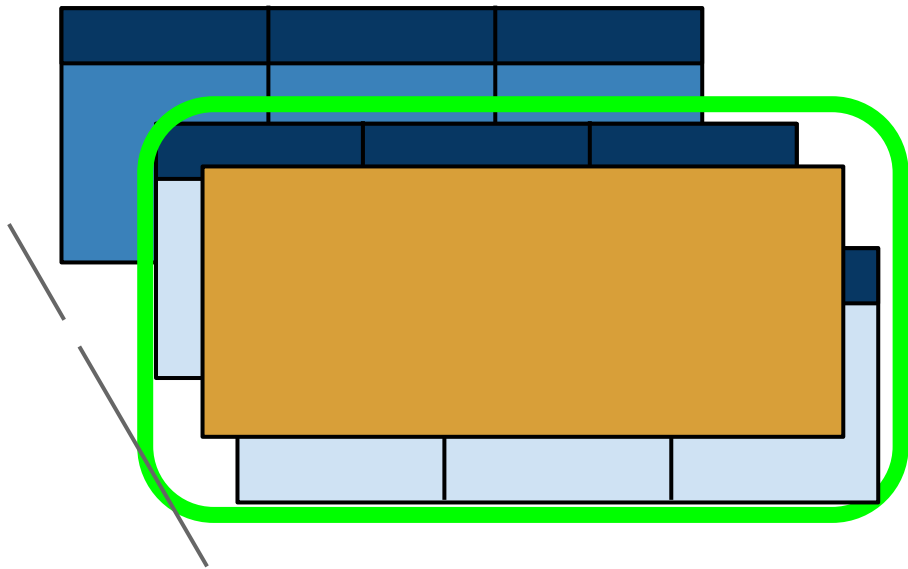
num = 2



# map



# mapPartitions



# na

Bob	Carol	0.2

Carol	null	0.3

Bob	Carol	null

from	to	amt
null	Bob	0.1

DataFrameNaFunctions  
Object

with methods: drop, fill, replace

# orderBy

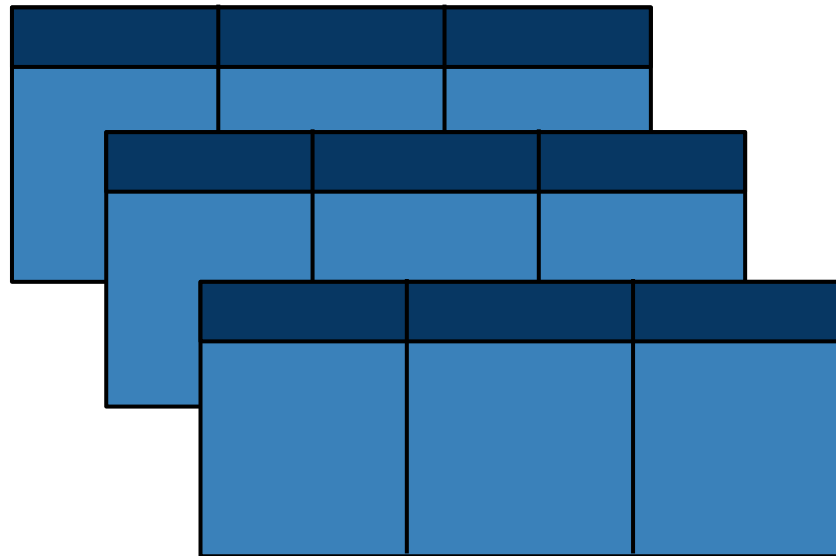
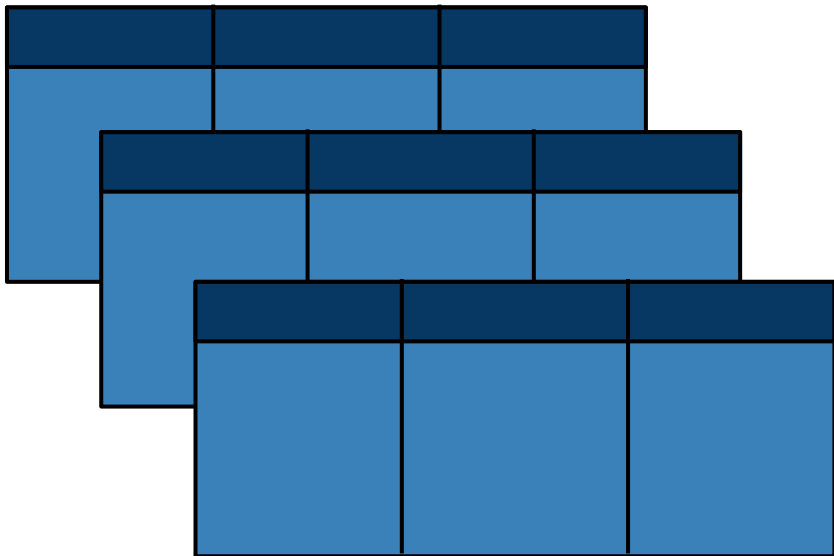
```
cols = ['from'], ascending = [False]
```

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Alice	Bob	0.1	
	Bob	Carol	0.2
	from	to	amt
	Carol	Dave	0.3

# persist

```
storageLevel =  
StorageLevel(MEMORY_ONLY_SER)
```



# printSchema

Carol	Dave	0.3

Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

stdout

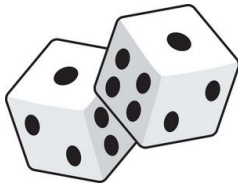
root

|-- from: string (nullable = true)  
|-- to: string (nullable = true)  
|-- amt: double (nullable = true)



# randomSplit

weights = [0.5,0.5]

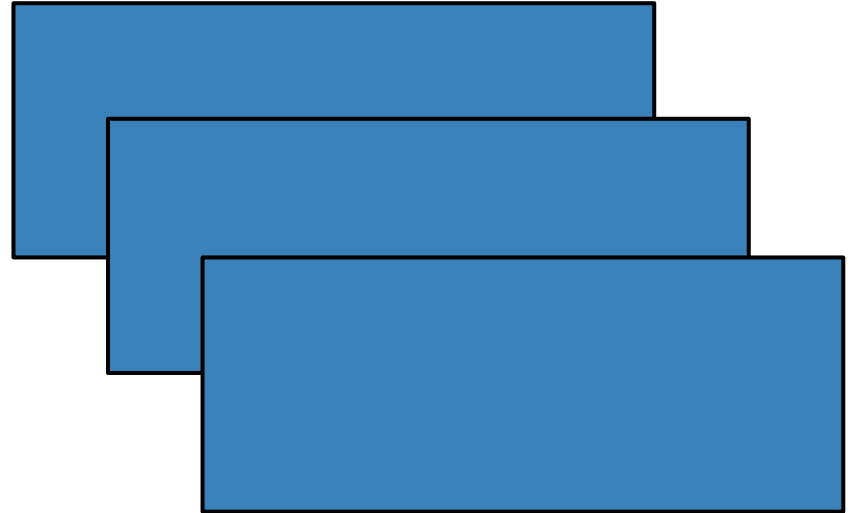
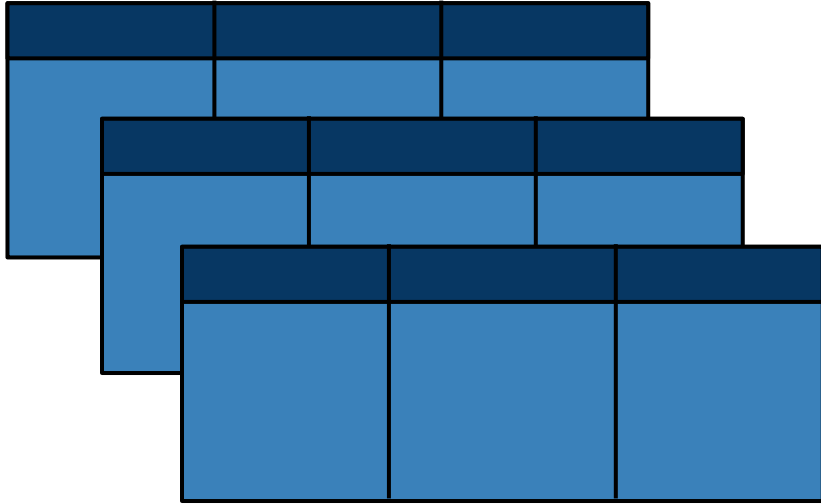


Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	Dave	0.3
from	to	amt
Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

# rdd



# registerTempTable

name = "TRANSACTIONS"

The diagram shows three tables stacked vertically, representing a temporary table structure. Each table has a dark blue header row and three light blue data rows. The tables are offset to the right, creating a cascading effect.







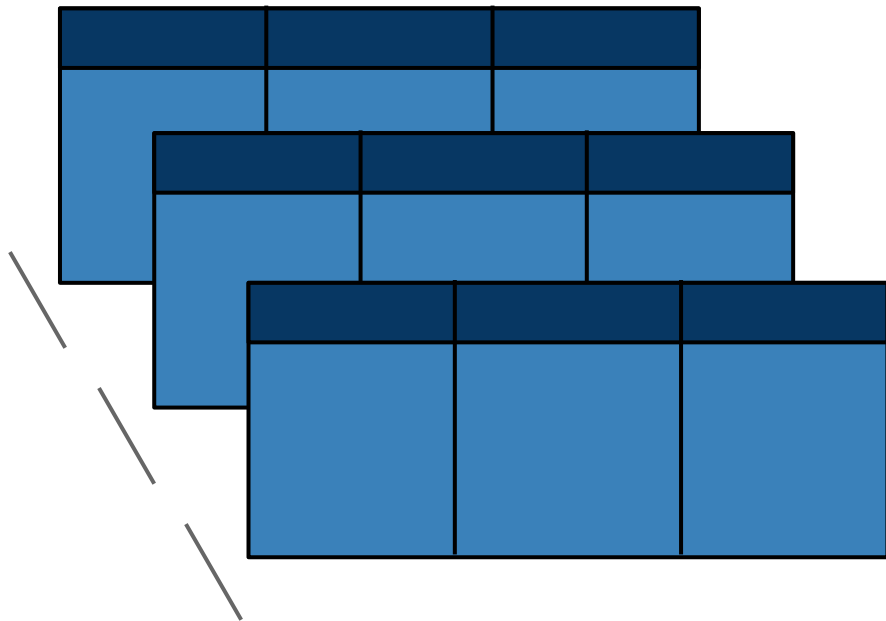
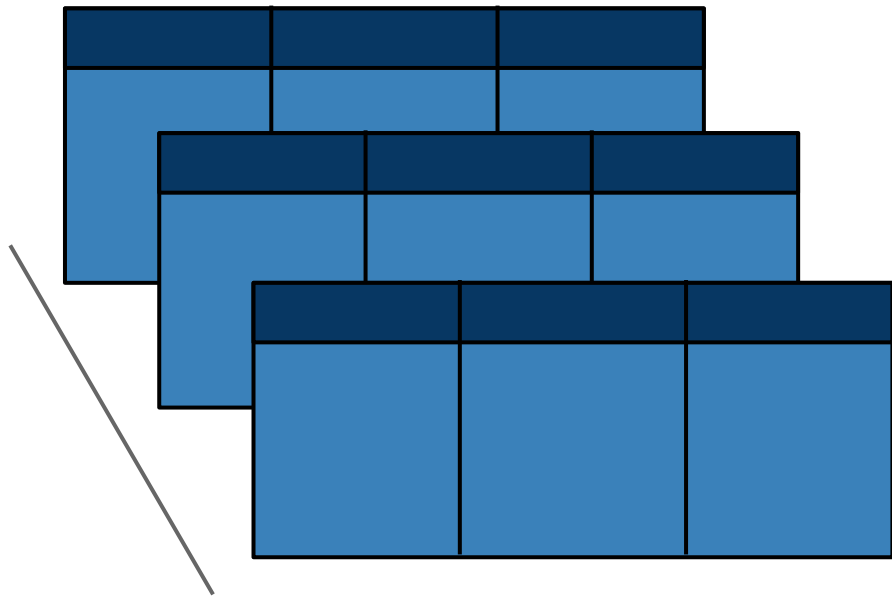




TRANSACTIONS

The diagram shows three tables stacked vertically, representing a permanent table structure. Each table has a dark blue header row and three light blue data rows. The tables are offset to the right, creating a cascading effect.

# repartition



# replace

```
to_replace = 'Dave' value = 'David'
```

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	David	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

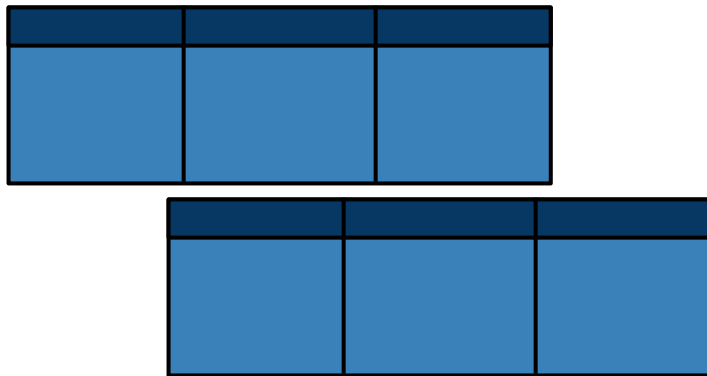
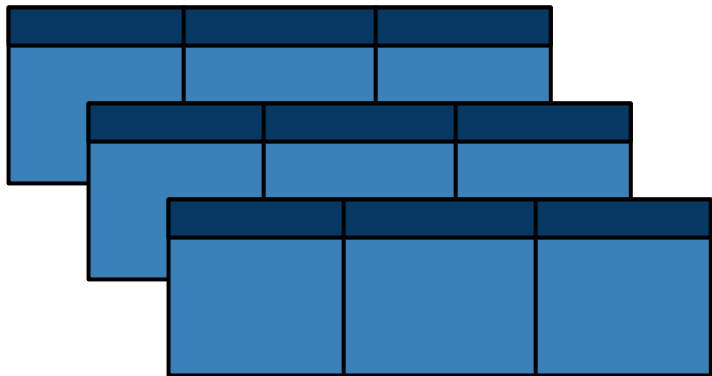
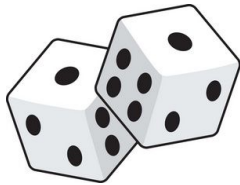
# rollup

cols = ['from', 'to']

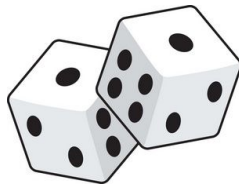
Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

null	null	
Carol	null	
Bob	null	
Carol	Dave	
Alice	null	
Bob	Carol	
from	to	agg(amt)
Alice	Bob	

# sample



# sampleBy



Bob	Carol	0.6
Bob	Bob	0.5
Alice	Dave	0.4
Alice	Alice	0.3
Alice	Carol	0.2
from	to	amt
Alice	Bob	0.1

Bob	Carol	0.6
Bob	Bob	0.5
from	to	amt
Alice	Bob	0.3



# schema

from	to	amt

from	to	amt
------	----	-----

# select

```
cols = ['from', 'amt']
```

from	to	amt

from	amt

# selectExpr

```
expr = ["substr(from,1,1)", "amt + 10"]
```

Carol	Dave	0.3

Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

The diagram illustrates three overlapping tables. The top table has columns *C* and 10.3. The middle table has columns *B* and 10.2. The bottom table has columns *from* and *amt*, with values *A* and 10.1 respectively.

# show

Carol	Dave	0.3

Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

stdout

```
+-----+-----+---+  
| from|  to|amt|  
+-----+-----+---+  
| Alice| Bob|0.1|  
| Bob|Carol|0.2|  
| Carol| Dave|0.3|  
+-----+-----+---+
```

# sort

cols = ['to']

Carol	Alice	0.3	
Bob	Carol	0.2	
	from	to	amt
	Alice	Bob	0.1

Bob	Carol	0.2

Alice	Bob	0.1

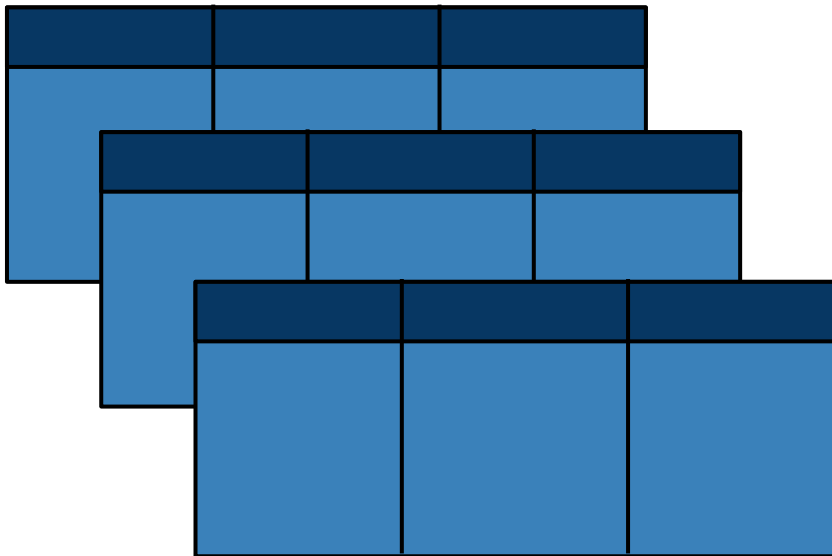
from	to	amt
Carol	Alice	0.3

# sortWithinPartitions

Carol	Alice	0.3	2
Bob	Carol	0.2	2
from	to	amt	p_id
Alice	Bob	0.1	1

Bob	Carol	0.2	2
Carol	Alice	0.3	2
from	to	amt	p_id
Alice	Bob	0.1	1

# stat



DataFrameStatFunctions  
Object  
with methods: `corr`, `cov`,  
`corsstab`, `freqItems`, `sampleBy`

# subtract

Carol	Dave	0.3
Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

Carol	Dave	0.1

Bob	Carol	0.2

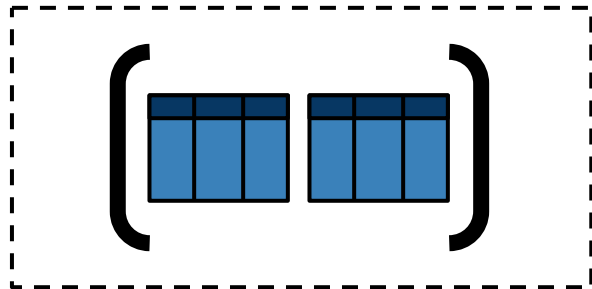
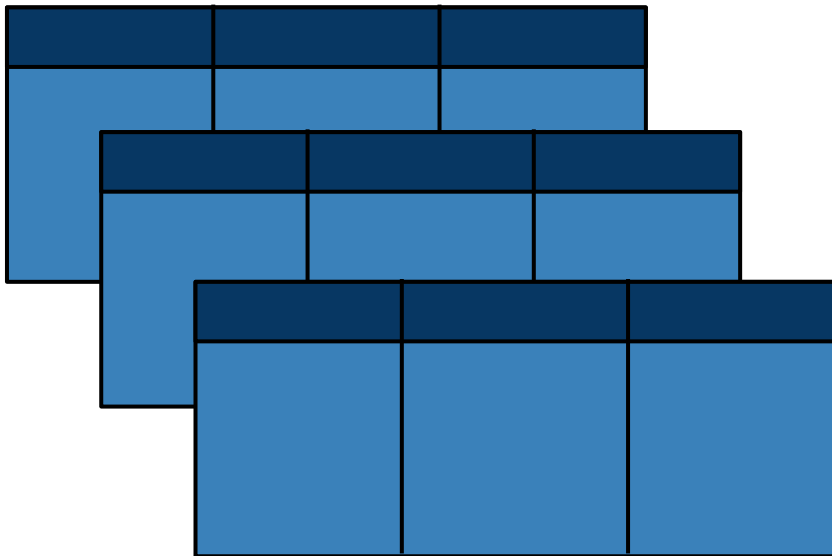
from	to	amt
Alice	Bob	0.1
	Carol	0.2

from	to	amt
Carol	Dave	0.3



# take

num = 2



# toDF

```
cols = ["seller", "buyer"]
```

from	to	amt

seller	buyer	amt

# toJSON

Carol	Alice	0.3

Bob	Carol	0.2

from	to	amt
Alice	Bob	0.1

u'{"from":"Carol","to":"Alice","amt":0.3}'

u'{"from":"Bob","to":"Carol","amt":0.2}'

u'{"from":"Alice","to":"Bob","amt":0.1}'

# toPandas

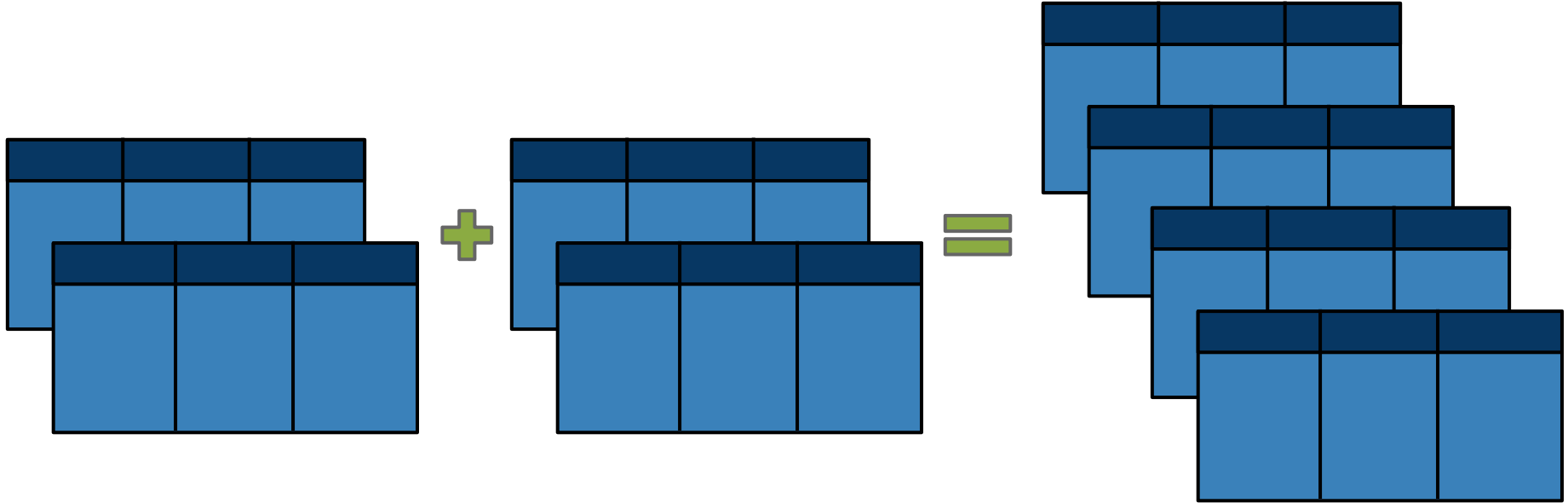
Carol	Alice	0.3

Bob	Carol	0.2

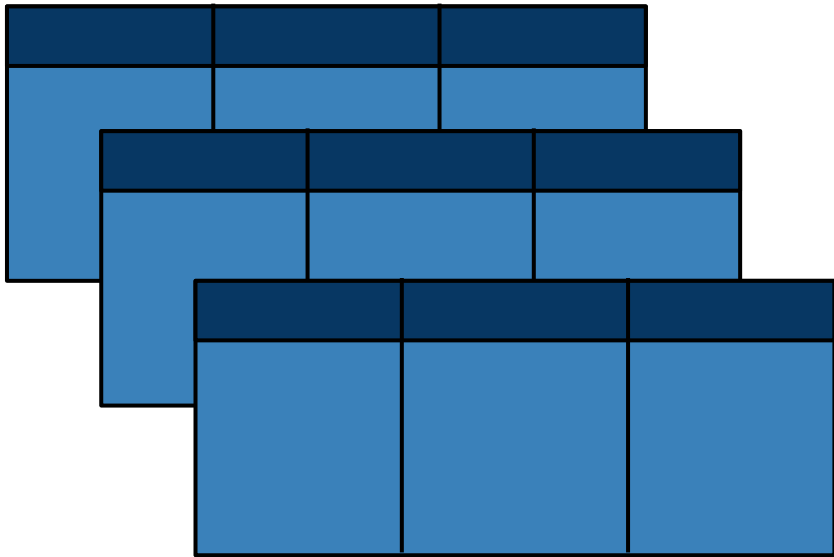
from	to	amt
Alice	Bob	0.1

	from	to	amt
0	Alice	Bob	0.1
1	Bob	Carol	0.2
2	Carol	Alice	0.3

# unionAll



# unpersist



# where (filter)

condition = "amt > 0.1"

Carol	Dave	0.3
Bob	Carol	0.2
from	to	amt
Alice	Bob	0.1

Carol	Dave	0.3
from	to	amt
Bob	Carol	0.2

# withColumn

colName = 'conf'

from	to	amt

from	to	amt	conf



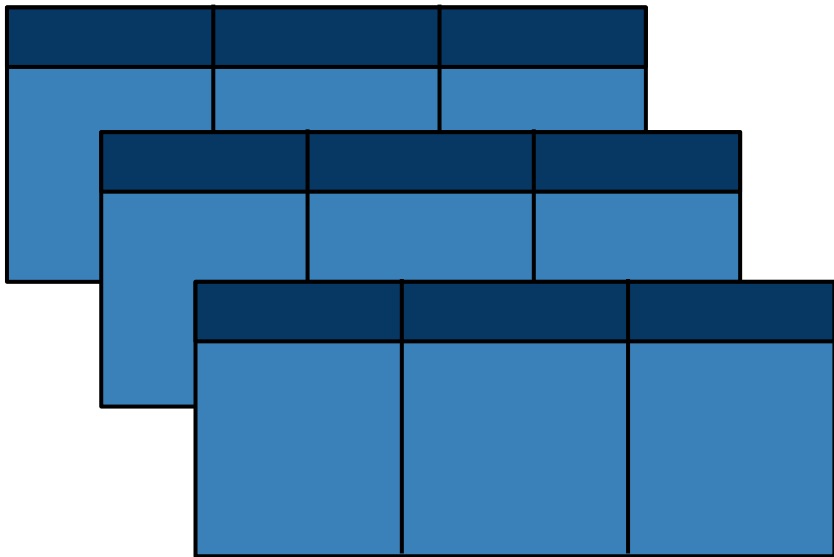
# withColumnRenamed

```
existing = 'amt' col = 'amount'
```

from	to	amt

from	to	amount

# write



DataFrameWriter  
Object

with methods: format,  
insertInto, jdbc, json, mode,  
option, options, orc, parquet,  
partitionBy, save, saveAsTable,  
text

PLEASE RESIST  
TRUMP