Aggregate on datasets

By aggregating data we can answer the following questions

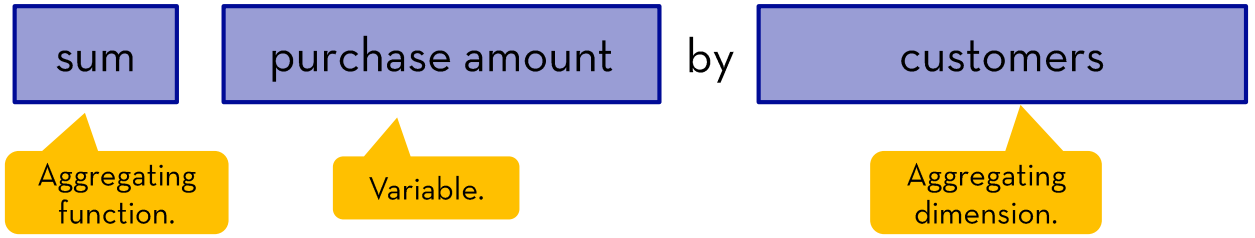
- What is the total value of purchases in 2015?
- How much has each customer spent per month?
- What is the average age of all male customers?

Aggregating means “do <<function>> to <<variable>> by <<dimension>>”

Aggregating has 2 components:

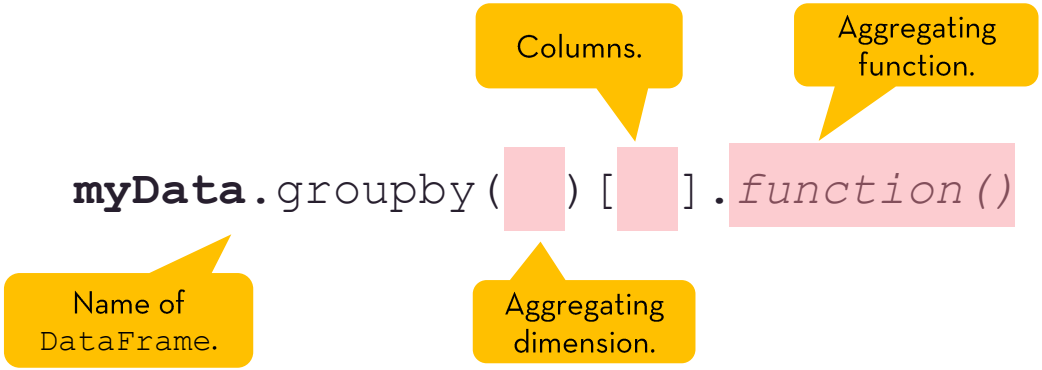
- Function and variable by which to aggregate.
- Dimension by which to aggregate.

For example:



General command structure for aggregating DataFrame objects on one dimension

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...



There are multiple ways of aggregating

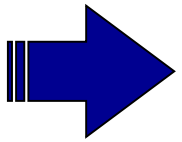
1. Apply an aggregating function to a variable by an aggregating dimension.
2. Apply an aggregating function to multiple variables by an aggregating dimension.
3. Apply an aggregating function to a variable by multiple aggregating dimensions.
4. Apply an aggregating function to a variable by an aggregating dimension to a selection of rows.
5. Apply an aggregating function to a variable by an aggregating dimension and select columns.

Apply an aggregating function to a variable by an aggregating dimension (1/2)

Option 1: `Grouby( )` with direct aggregation procedure

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum PurchAmount by Customer



Customer	
100001	279.90
100002	499.95
100003	379.90
100004	499.95
...	...
149236	119.90

Now summed.

Name of **`DataFrame`**.

Aggregating dimension.

Result is a Multiindex by default, use option `as_index` to stick to single index.

```
myData.groupby("Customer", as_index=False)
      ["PurchAmount"].sum()
```

Columns.

Aggregating function.

Apply an aggregating function to a variable by an aggregating dimension (2/2)

Option 2: `agg( )` `including renaming`

Using `agg( )` enables multiple and different aggregation functions on columns (see later).

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum PurchAmount by Customer and rename it "AggPurch"

Customer	AggPurch
100001	279.90
100002	499.95
100003	379.90
100004	499.95
...	...
149236	119.90

"`Agg( )`" allows to specify a name here.

```
myData.groupby("Customer", as_index=False)["PurchAmount"].agg(["sum"]).rename(columns={"PurchAmount":"AggPurch"})
```

Use the aggregation-function `agg()`.

Aggregating function is "`sum`".

Name of new column in the output.

Apply multiple aggregating functions to a variable by an aggregating dimension

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum and select max of PurchAmount by Customer

Customer	AggPurch	Purch_max
100001	279.90	199.95
100002	499.95	499.95
100003	379.90	249.95
100004	499.95	499.95
...	...	...
149236	119.90	499.95

Aggregating dimension.

```
myData.groupby("Customer", as_index=False)["PurchAmount"].agg(
        {"AggPurch": "sum", "Purch_max": "max"})
```

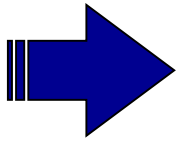
First aggregation function.

Second aggregation function.

Apply an aggregating function to the whole dataset

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum all PurchAmounts



```
419.80
```

Returns a number.

Sum all purchase amounts.

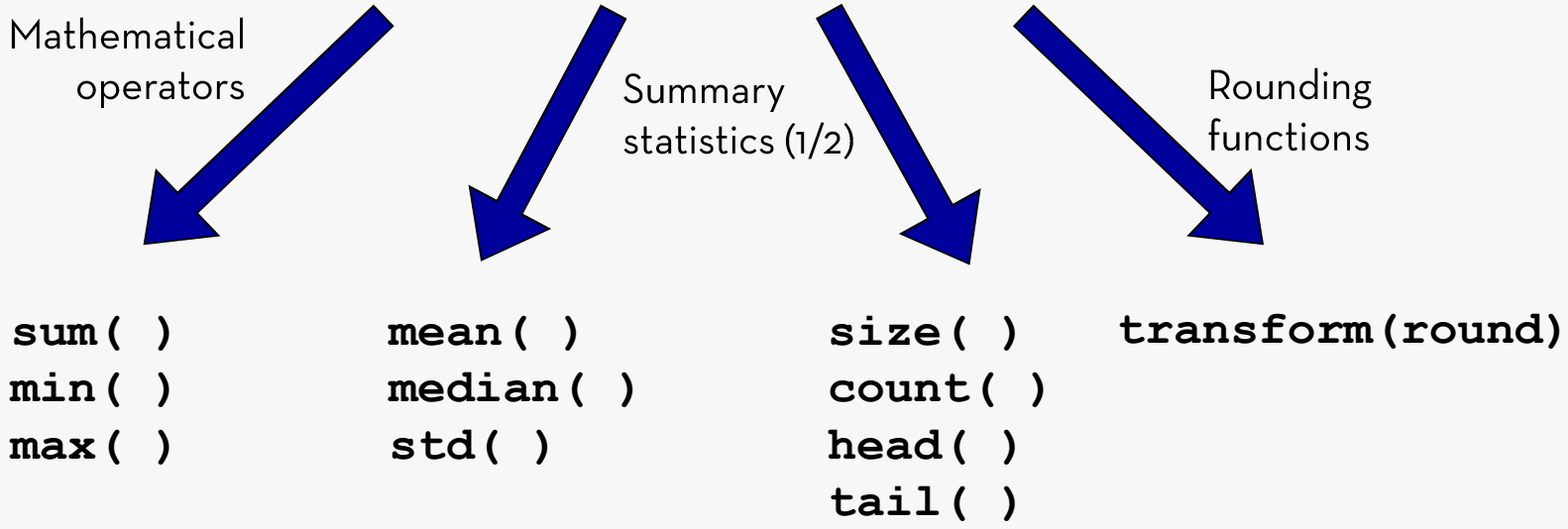
```
myData["PurchAmount"].sum()
```

Name of DataFrame.

Aggregating dimension is implicit and does not need to be specified.

Python Basics: a small list of aggregating functions

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...



Sidenote: Create new columns in the original DataFrame with the transform()-function

```
myData["AggPurch"]= myData.groupby("Customer")
                          ["PurchAmount"].transform(sum)
```

Name of the new column.

Replicates entry for every row in the DataFrame.

Creates a new column in **myData**.

Customer sequence is not changed anymore.

Customer	TransDate	Quantity	PurchAmount	Cost	AggPurch
149332	15.11.2005	1	199.95	107.00	274.85
172951	29.08.2008	1	199.95	108.00	889.80
120621	19.10.2007	1	99.95	49.00	99.95
149236	14.11.2005	1	39.95	18.95	119.90
149236	12.06.2007	1	79.95	35.00	119.90
...	...	...	...	...	...

Apply a/multiple aggregating function(s) to multiple variables by an aggregating dimension

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Apply various aggregation functions to PurchAmount and Quantity aggregated by Customer

	Purch Amount		Quantity
Customer	max	sum	sum
100001	199.95	279.90	2
100002	499.95	499.95	1
100003	249.95	379.90	2
100004	499.95	499.95	1
...	...	...	
149236	79.95	119.90	2

Sum all purchases AND quantities.

Pass a list to apply multiple aggregation functions.

```
myData.groupby("Customer")
      .agg({"PurchAmount":["max", "sum"],
            "Quantity":"sum"})
```

Use original variable names only.

Apply an aggregating function to multiple variables by multiple aggregating dimensions

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum PurchAmount and sum Quantity aggregated by Customer and TransDate

Customer	TransDate	PurchAmount	Quantity
100001	2011-06-25	79.95	1
	2011-08-24	199.95	1
100002	2004-12-29	499.95	1
100003	2012-01-23	379.90	2
...		...	

Multiple aggregating dimensions.

```
myData.groupby(["Customer","TransDate"])
                    ["PurchAmount","Quantity"].sum()
```

Apply an aggregating function to a variable by an aggregating dimension to a selection of rows

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select rows 2 to 5 and sum PurchAmount by Customer

Customer	PurchAmount
120621	99.95
149236	119.90
172951	199.95

Returns a DataFrame.

```
myData.iloc[1:6].groupby("Customer", as_index=False)
                ["PurchAmount"].sum()
```

Select rows.

Exercise
Aggregate on datasets

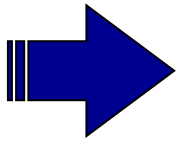
1. Sum `PurchAmount` by `Customer` and `TransDate`.
2. Count number of transactions by `Customer`.

Advanced aggregating topics

Aggregate a variable by a transformed aggregating dimension

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum PurchAmount by month of each year



TransDate	
2004-12	27623.90
2005-01	83363.73
2005-02	87341.59
...	...

Command part of **datetime**.

```
myData.groupby(myData["TransDate"].dt.to_period("M"))
      ["PurchAmount"].sum( )
```

Combines period and month, e.g. "2012-09".

Sidenote: Chaining saves memory and is faster

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum PurchAmount by Customer and select Customers with aggregated sums greater than 100

Customer	AggPurch
100001	279.90
100002	499.95
...	...

Customer	AggPurch
149332	199.95
172951	199.95
...	...

```
myData.groupby("Customer")["PurchAmount"].sum()
      [lambda x: x >= 100]
```

Use "[" and "]" for chaining the output created before.

Select all observations where the **aggregated purchase sum per customer** is greater then 100.

The same as:

```
myData_agg = myData.groupby("Customer")
             ["PurchAmount"].sum()
myData_agg.loc[myData_agg>=100,]
```

Sidenote: Pay attention to operation sequences

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Sum PurchAmount by Customer and select Customers with aggregated sums greater than 100

Customer	AggPurch
100001	279.90
100002	499.95
...	...

Customer	AggPurch
149332	199.95
172951	199.95
...	...

```
myData.groupby("Customer")["PurchAmount"].sum()
                          [lambda x: x >= 100]
```

⚠ Not the same as:

```
myData.loc[myData["PurchAmount"]>100,].
groupby("Customer")["PurchAmount"].sum()
```

Exercise
Advanced aggregating topics

1. Aggregate the purchase amount (sum) of all transaction per customer on a yearly basis for year 2007 and 2008.
2. How many customers purchased for more than 50$ in total between 2008 and 2009?

Selecting using an aggregating dimension

Select the first 3 purchases of each customer

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
172951	29.08.2008	1	249.95	162.50	128888288
172951	29.08.2008	1	39.95	18.95	128888288
172951	28.01.2009	1	79.95	35.00	129113804
...	...	...	...	...	...

Select the first 3 purchases of each customer

Customer	TransDate	Quantity	Purch Amount	Cost
149332	15.11.2005	1	199.95	107.00
172951	29.08.2008	1	199.95	108.00
172951	29.08.2008	1	249.95	162.50
172951	29.08.2008	1	39.95	18.95
149236	12.06.2007	1	79.95	35.00
...	...	...	...	...

```
myData.groupby("Customer").head(3).sort("Customer")
```

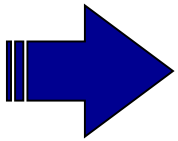
Select the first three entries of every customer.

Optional: Sort by ascending order of "Customer".

Select the last purchase of each customer

Customer	Trans Date	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
172951	29.08.2008	1	249.95	162.50	128888288
172951	29.08.2008	1	39.95	18.95	128888288
172951	28.01.2009	1	79.95	35.00	129113804
...	...	...	...	...	...

Select the last purchase of each customer



Customer	Trans Date	Quantity	Purch Amount	Cost
172951	28.01.2009	1	79.95	35.00
...	...	...	...	...

```
myData.groupby("Customer").tail(1).sort("Customer")
```

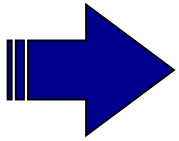
Select the last entry for every customer.

Optional: Sort by ascending order of "Customer".

Updating columns using an aggregating dimension

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Add a column counting the quantities per Customer



Customer	TransDate	...	Count
149332	15.11.2005		3
172951	29.08.2008		4
120621	19.10.2007		1
149236	14.11.2005		2
149236	12.06.2007		2
...	...	...	...

```
myData["Count"]= myData.groupby("Customer")["Customer"]
                 .transform("count")
```

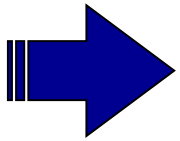
Use transform to save output in a new column.

Counts the total number of occurrences per customer.

Updating columns using an aggregating dimension

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Add a column counting the transactions per Customer



Customer	TransDate	...	RelDate
149332	15.11.2005		1
172951	29.08.2008		1
120621	19.10.2007		1
149236	14.11.2005		1
149236	12.06.2007		2
...	...	...	...

```
myData["RelDate"]=myData.groupby("Customer").cumcount()+1
```

Save output as variable "RelDate" into the **DataFrame**.

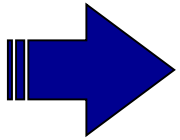
Counts the number of transactions per customer from 0 to transactions-1.

Add +1 to correct for counting from 0 onwards.

Creating a lagged variable

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Add a column with lagged cost by one observation aggregated by customer



Customer	TransDate	...	Cost	CostLag
149332	15.11.2005	1	107.00	NA
172951	29.08.2008		108.00	NA
120621	19.10.2007		49.00	NA
149236	14.11.2005		18.95	NA
149236	12.06.2007		35.00	18.95
...	...	...	...	...

Only works for customers with more than two purchases.

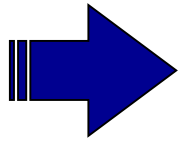
```
myData["CostLag"]=myData.groupby("Customer")["Cost"].shift(periods=1)
```

Lag.

Cumulating variables

Customer	TransDate	Quantity	Purch Amount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Add a column with cumulated cost aggregated by customer



Customer	TransDate	...	Cost	totSpend
149332	15.11.2005		107.00	107.00
172951	29.08.2008		108.00	108.00
120621	19.10.2007		49.00	49.00
149236	14.11.2005		18.95	18.95
149236	12.06.2007		35.00	53.95
...	...	...	...	...

```
myData["totSpend"] = myData.groupby("Customer")["Cost"].cumsum()
```

Save output as variable "totSpend" into the DataFrame.

Variable of interest.

Cumulates the "Cost" variable.

Exercise
Selecting using an aggregating dimension

1. Add a column to `myData` with the total number of purchases per customers.
2. Create a lead shifted variable for `TransDate` (by one period) by customer.