CHAPTER 4: DATA TRANSFORMATION & ANALYSIS

DR. MOHD KHAIRUL BAZLI MOHD AZIZ

PUSAT SAINS MATEMATIK, UMP

CONTENT

3.1 SUBSETTING, FILTERING, AND GROUPING

Data Subsetting

- Test data subsetting is extracting a smaller sized – referential intact – set of data from a 'production' database to a non-production environment. Many customers ask us: "How should we create a subset?" and "How usable is a subset compared to my copy of a production database?" The concept of data subsetting is surprisingly simple: take a consistent part of a database and transfer it to another database. That's all. Of course, the actual data subsetting isn't that simple. Especially selecting the right data for the job is tricky, whether it's testing or development. Why? Because you need to filter data. The complexity is in getting all the right data to create a consistent dataset over all tables that also fulfills the tester's needs.

DATA SUBSETTING

- Without question, most organizations don't need all the data they have stored in their non-production environment and it's costing them money.

- With the use of subsets:
  1. The need for data storage is decreased (sometimes by more than 90%)
  2. Idle times are significantly reduced
  3. High control in test and development turnarounds
  4. Developers influence the data they need

DATA FILTERING

- Data filtering is the task of reducing the content of noise or errors from measured process data. It is an important task because measurement noise masks the important features in the data and limits their usefulness in practice. Various techniques have been developed to filter process data, and include model-free techniques, model-based techniques, and techniques based on empirical models.

DATA FILTERING

- Filtering may be used to:
  1. Look at results for a particular period of time.
  2. Calculate results for particular groups of interest.
  3. Exclude erroneous or "bad" observations from an analysis.
  4. Train and validate statistical models.

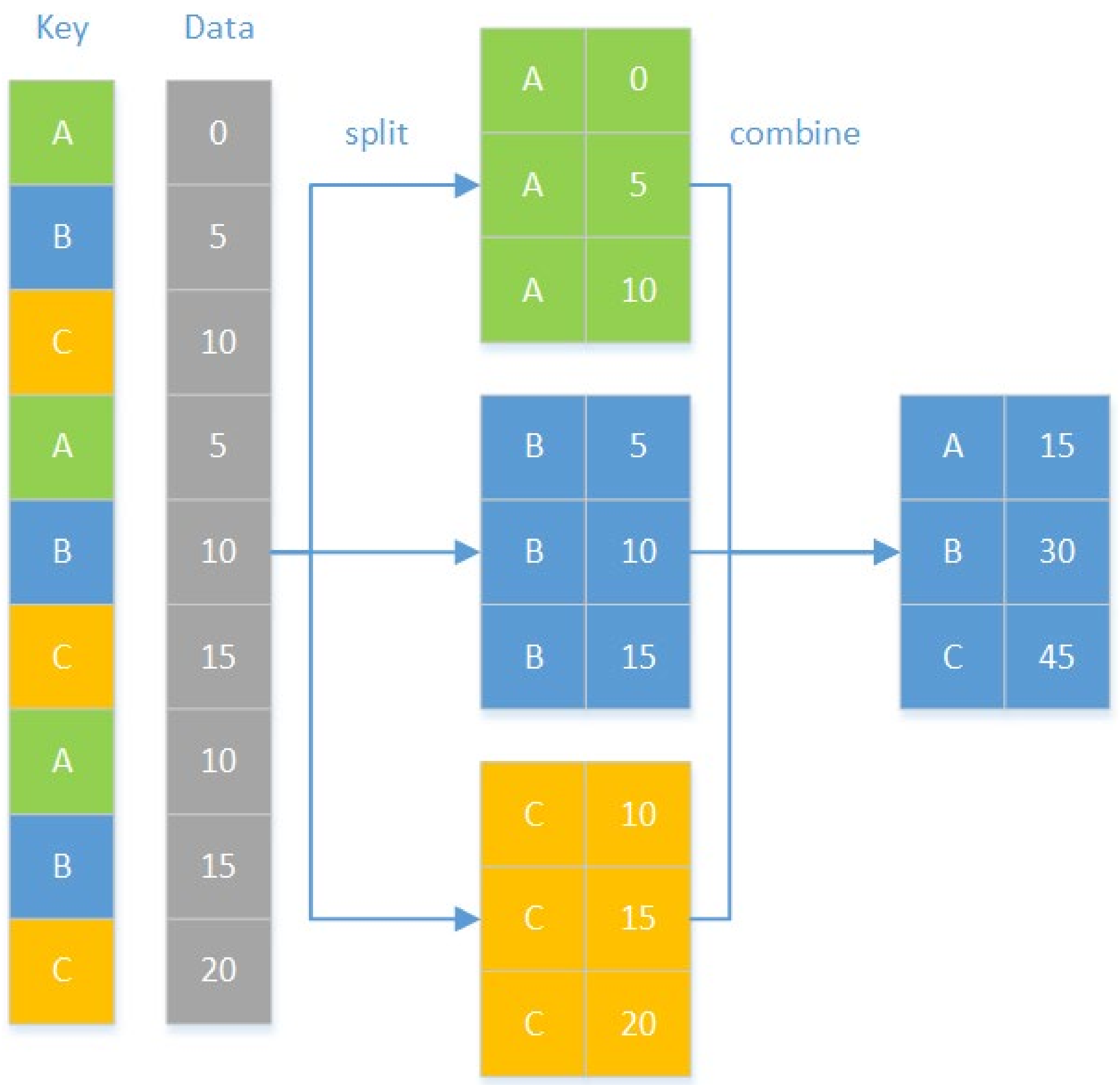
- Filtering requires you to specify a rule or logic to identify the cases you want to included in your analysis. Filtering can also be referred to as “subsetting” data, or a data “drill-down”. In this article we illustrate a filtered data set and discuss how you might use filtering.

DATA GROUPING

- Grouping of data plays a significant role when we have to deal with large data. This information can also be displayed using a pictograph or a bar graph. Data formed by arranging individual observations of a variable into groups, so that a frequency distribution table of these groups provides a convenient way of summarizing or analyzing the data is termed as grouped data.
- Pandas has groupby function to be able to handle most of the grouping tasks conveniently. But there are certain tasks that the function finds it hard to manage.

DATA GROUPING

- groupby is one of the most important Pandas functions. It is used to group and summarize records according to the split-apply-combine strategy. The following diagram shows the workflow:



DATA GROUPING

- GroupBy refers to a process involving one or more of the following steps:
  1. Splitting the data into groups based on some criteria
  2. Applying a function to each group independently
  3. Combining the results into a data structure

DATA GROUPING

- In many situations, we can split the dataset into groups and do something with those groups. In the apply step, we may wish to do one of the following:

1. Aggregation: Compute a summary statistic (or statistics) for each group – sum, mean, and so on
2. Transformation: Perform a group-specific computation and return a like indexed object – z-transformation or filling missing data with a value
3. Filtration: Discard a few groups, according to a group-wise computation that evaluates TRUE or FALSE

4.2 CONCATENATING, MERGING, AND JOINING

- Merging and joining tables or datasets are highly common operations in the day-today job of a data wrangling professional. Often, the key data is present in multiple tables, and those records need to be brought into one combined table that matches on that common key. This is an extremely common operation in any type of sales or transactional data, and therefore must be mastered by a data wrangler.
- Three most important techniques for combining data in Pandas:

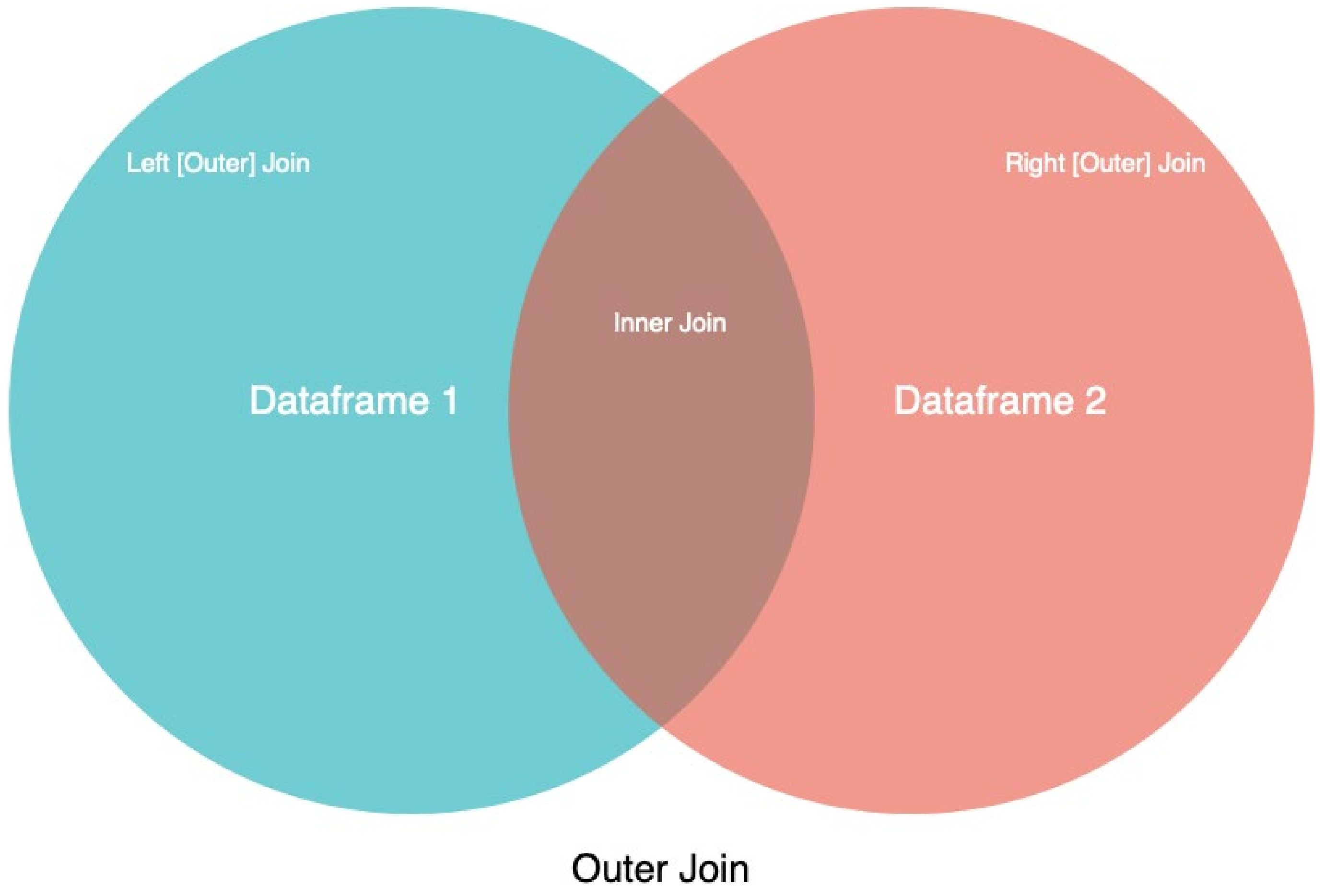
1. `merge()` for combining data on common columns or indices
2. `.join()` for combining data on a key column or an index
3. `concat()` for combining DataFrames across rows or columns

PANDAS `MERGE()` : COMBINING DATA ON COMMON COLUMNS OR INDICES

- You can use `merge()` any time you want to do database-like join operations. It's the most flexible of the three operations you'll learn.
- When you want to combine data objects based on one or more keys in a similar way to a relational database, `merge()` is the tool you need. More specifically, `merge()` is most useful when you want to combine rows that share data.

Before getting into the details of how to use merge(), you should first understand the various forms of joins:

- inner
- outer
- left
- right



PANDAS `.JOIN()`: COMBINING DATA ON A COLUMN OR INDEX

- While `merge()` is a module function, `.join()` is an object function that lives on your DataFrame. This enables you to specify only one DataFrame, which will join the DataFrame you call `.join()` on.
- Joining is performed based on index keys and is done by combining the columns of two potentially differently indexed DataFrames into a single one. It offers a faster way to accomplish merging by row indices. This is useful if the records in different tables are indexed differently but represent the same inherent data and you want to merge them into a single table.

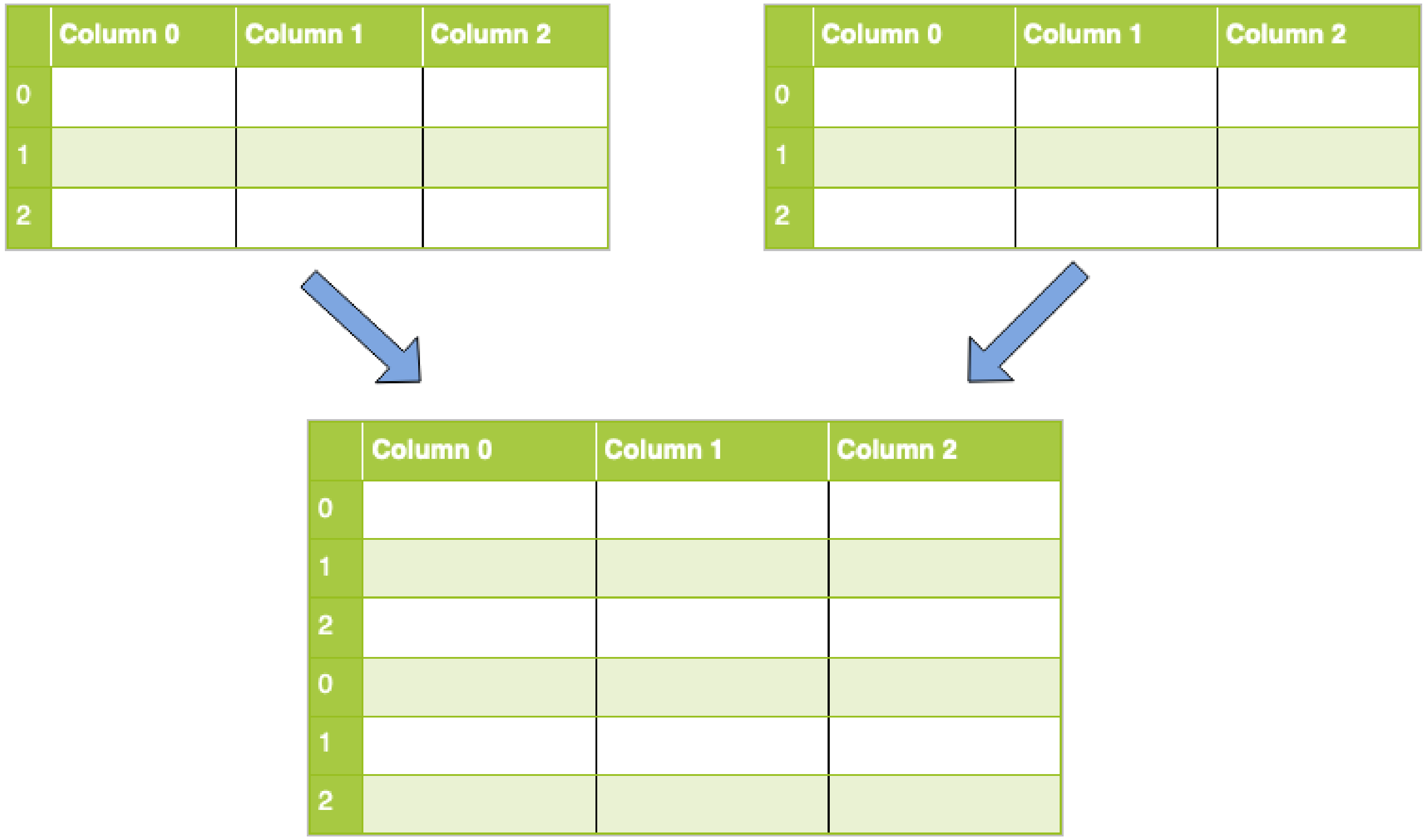
PANDAS `CONCAT()`: COMBINING DATA ACROSS ROWS OR COLUMNS

- Concatenation is a bit different from the merging techniques you saw above. With merging, you can expect the resulting dataset to have rows from the parent datasets mixed in together, often based on some commonality. Depending on the type of merge, you might also lose rows that don't have matches in the other dataset.

- With concatenation, your datasets are just stitched together along an axis — either the row axis or column axis.

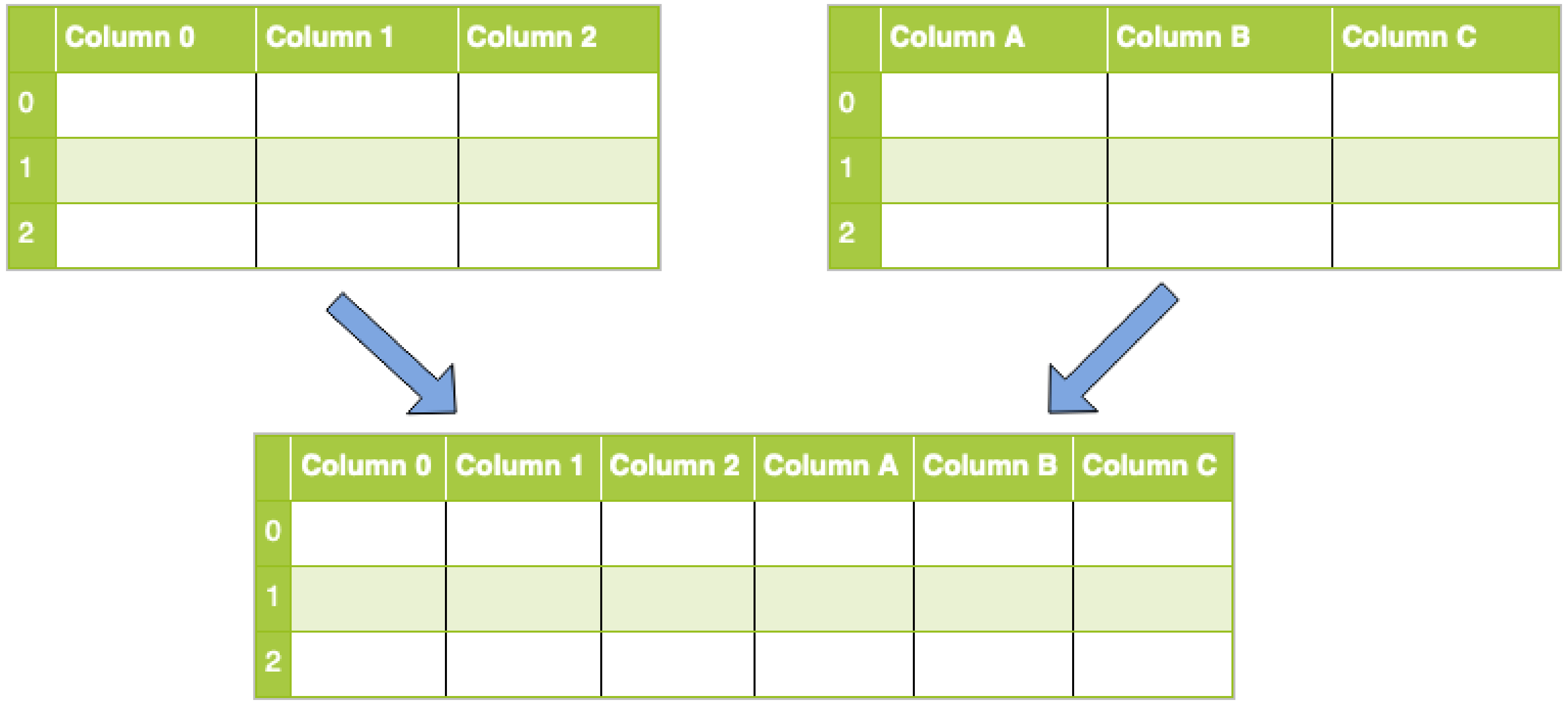
CONCATENATION

Visually, a concatenation with no parameters along rows would look like this:



CONCATENATION

What if instead you wanted to perform a concatenation along columns? First, take a look at a visual representation of this operation:



4.3 HIERARCHICAL INDEXING

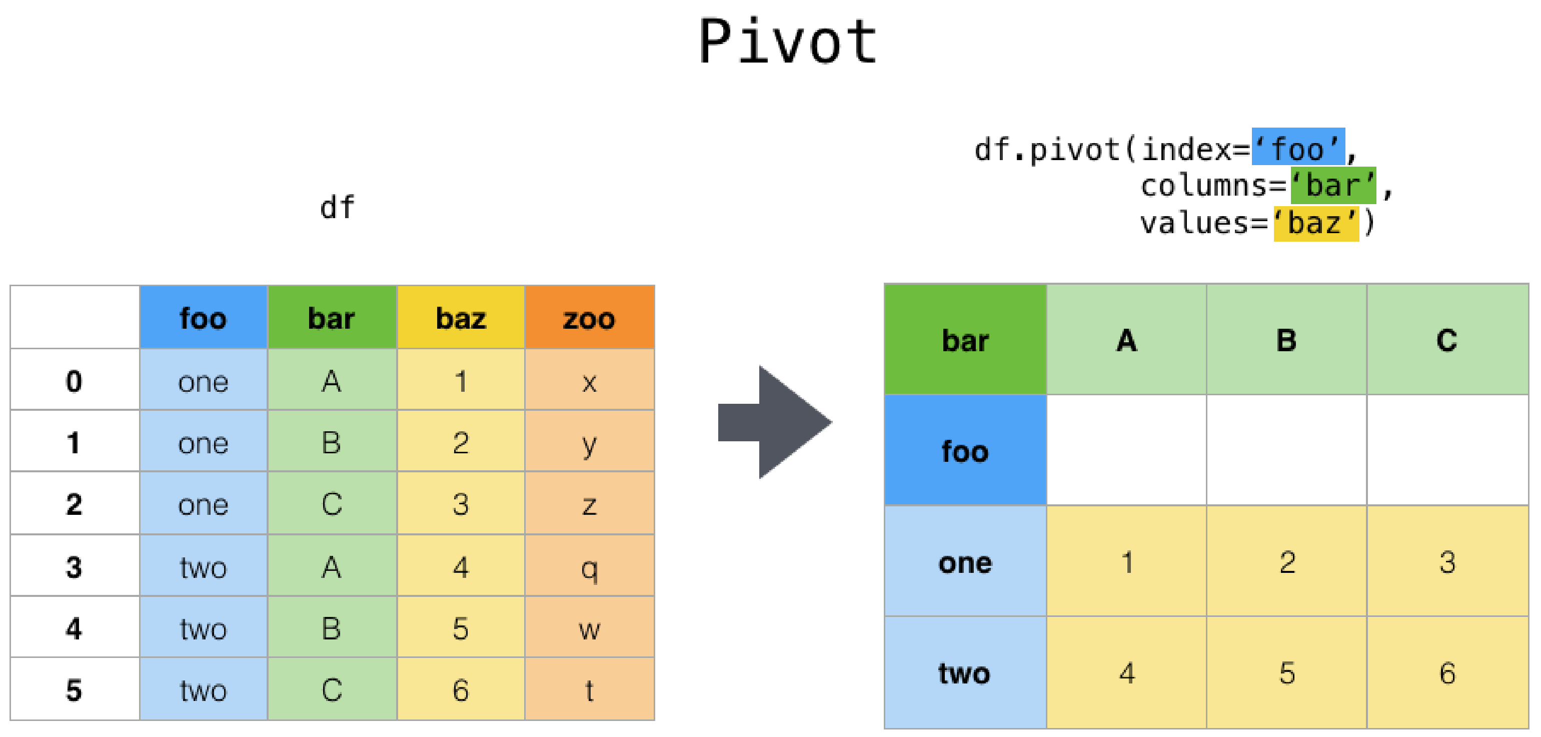
- Hierarchical indexing is an important feature of pandas that enable us to have multiple index levels. In this section, we will learn more about indexing and access to data with these indexing.
  1. Creating multiple index
  2. Partial indexing
  3. Unstack the data
  4. Column indexing
  5. Swap and sort level
  6. Summary statistics by level

4.4 RESHAPING AND PIVOTING

- In Pandas data reshaping means the transformation of the structure of a table or vector (i.e. DataFrame or Series) to make it suitable for further analysis. Some of Pandas reshaping capabilities do not readily exist in other environments (e.g. SQL or bare bone R) and can be tricky for a beginner.

PIVOT

The pivot function is used to create a new derived table out of a given one. Pivot takes 3 arguments with the following names: index, columns, and values. As a value for each of these parameters you need to specify a column name in the original table. Then the pivot function will create a new table, whose row and column indices are the unique values of the respective parameters. The cell values of the new table are taken from column given as the values parameter.



Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

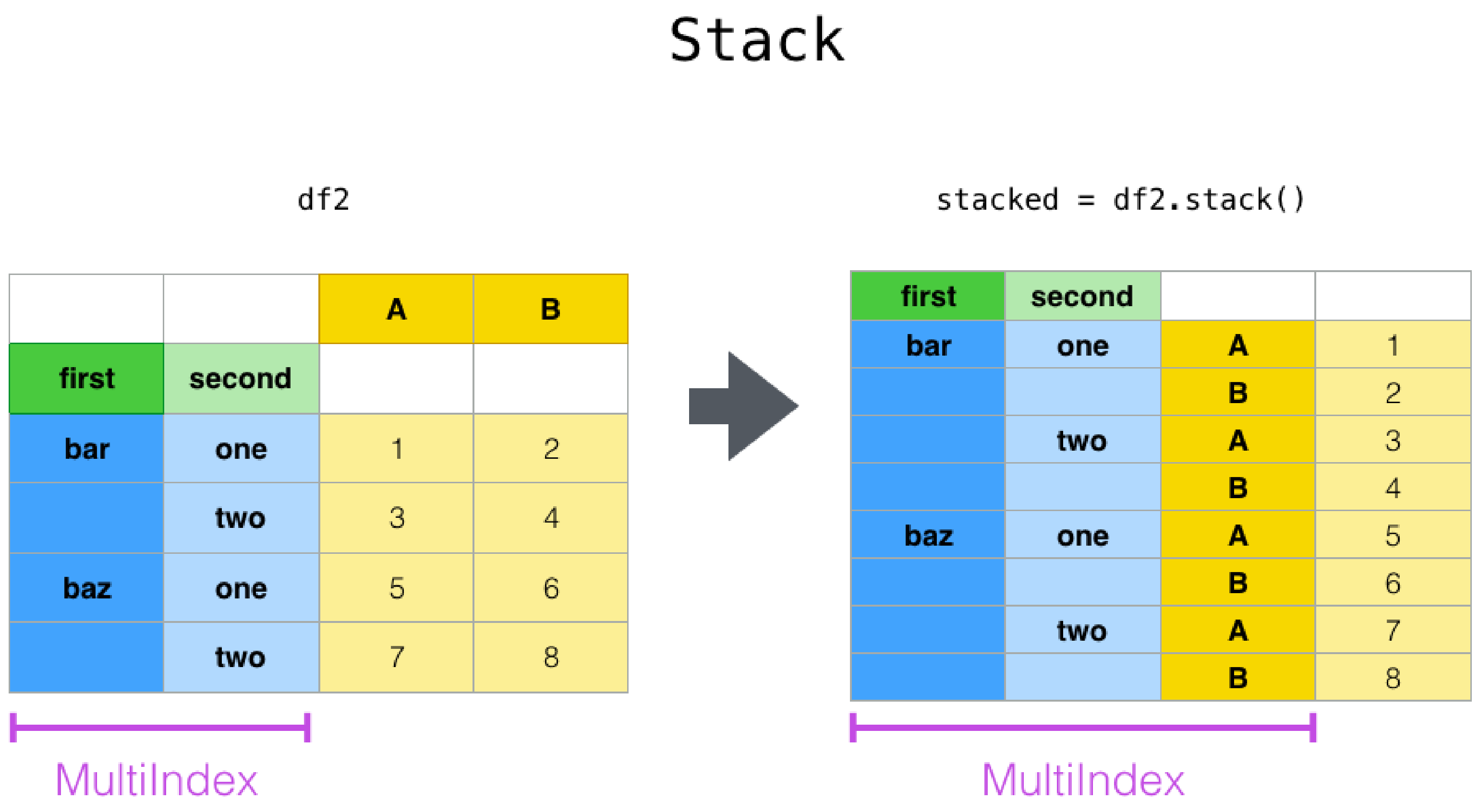
RESHAPING BY STACKING AND UNSTACKING

Closely related to the pivot() method are the related `stack()` and `unstack()` methods available on Series and DataFrame. These methods are designed to work together with MultiIndex objects (see the section on hierarchical indexing). Here are essentially what these methods do:

- `stack()`
- `unstack()`

STACK

`stack()`: "pivot" a level of the (possibly hierarchical) column labels, returning a DataFrame with an index with a new inner-most level of row labels.



Stack

`df2`

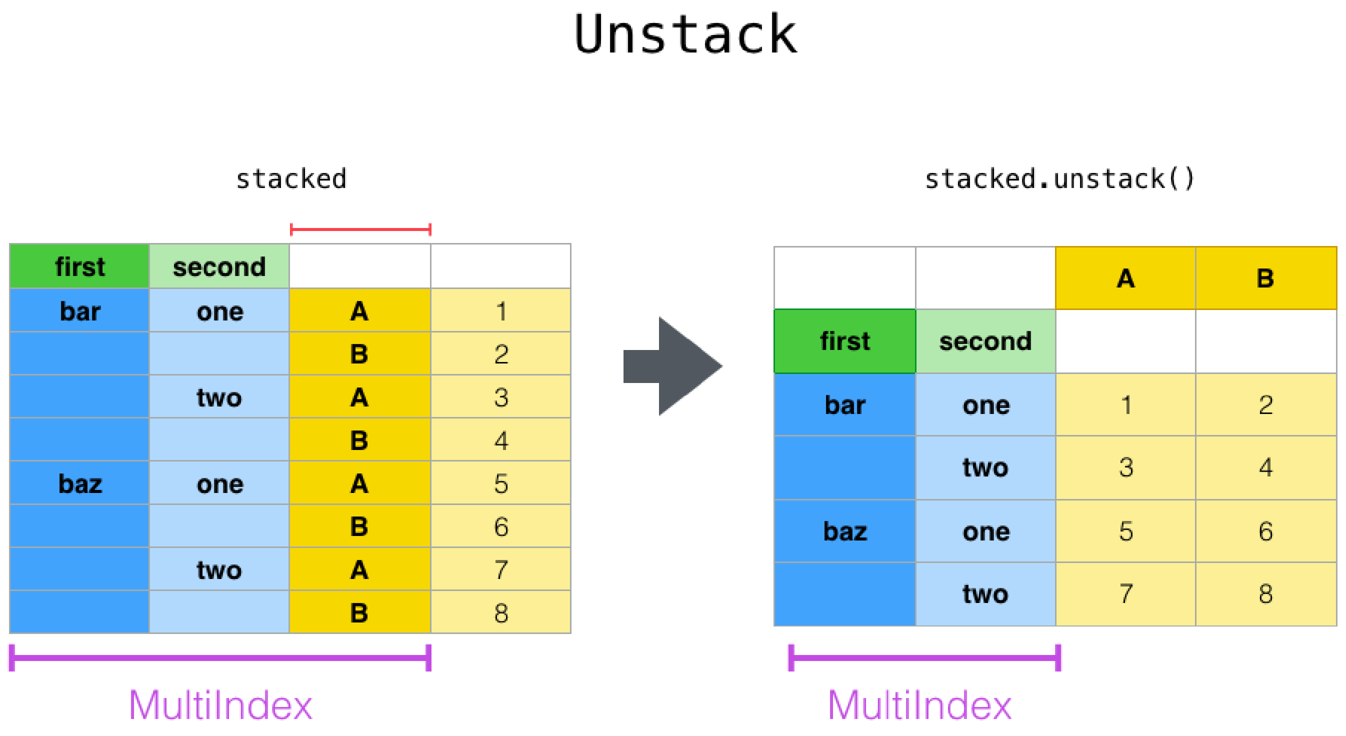
		A	B
first	second		
bar	one	1	2
	two	3	4
baz	one	5	6
	two	7	8

`stacked = df2.stack()`

first	second		
bar	one	A	1
		B	2
	two	A	3
		B	4
baz	one	A	5
		B	6
	two	A	7
		B	8

UNSTACK

`unstack()`: (inverse operation of stack()) "pivot" a level of the (possibly hierarchical) row index to the column axis, producing a reshaped DataFrame with a new inner-most level of column labels.



stacked

first	second		
bar	one	A	1
		B	2
	two	A	3
		B	4
baz	one	A	5
		B	6
	two	A	7
		B	8

stacked.unstack()

		A	B
first	second		
bar	one	1	2
	two	3	4
baz	one	5	6
	two	7	8

4.5 DATA AGGREGATION

- Data aggregation is the process where data is collected and presented in a summarized format for statistical analysis and to effectively achieve business objectives. Data aggregation is vital to data warehousing as it helps to make decisions based on vast amounts of raw data. Data aggregation provides the ability to forecast future trends and aids in predictive modeling. Effective data aggregation techniques help to minimize performance problems.
- Aggregation provides more information based on related clusters of data such as an individual's income or profession. For example, a store may want to look at the sales performance for different regions, so they would aggregate the sales data based on region.
- Queries with aggregation (with mathematical functions) provide faster results. For example, the query for the sum of sales of a product in a month brings up faster results than the query for sales of the product in general. This is because the aggregation is applied on the former query and only the sum is displayed, while the latter query brings up individual records. Faster queries imply the better performance of the system.

- Types of aggregation with mathematical functions:
  1. Sum – Adds together all the specified data to get a total.
  2. Average – Computes the average value of the specific data.
  3. Max – Displays the highest value for each category.
  4. Min – Displays the lowest value for each category.
  5. Count – Counts the total number of data entries for each category.
- Data can also be aggregated by date, allowing trends to be shown over a period of years, quarters, months, etc. These aggregations could be placed in a hierarchy, where you can view the data trends over a period of years, then see the data trends over months for each individual year.

4.6 PIVOT TABLES AND CROSS-TABULATION

- Pivot tables and crosstabs are ways to display and analyze sets of data. Both are similar to each other, with pivot tables having just a few added features.
- Pivot tables and crosstabs present data in tabular format, with rows and columns displaying certain data. This data can be aggregated as a sum, count, max, min, or average if desired. These tools allow the user to easily recognize trends, see relationships between their data, and access information quickly and efficiently.

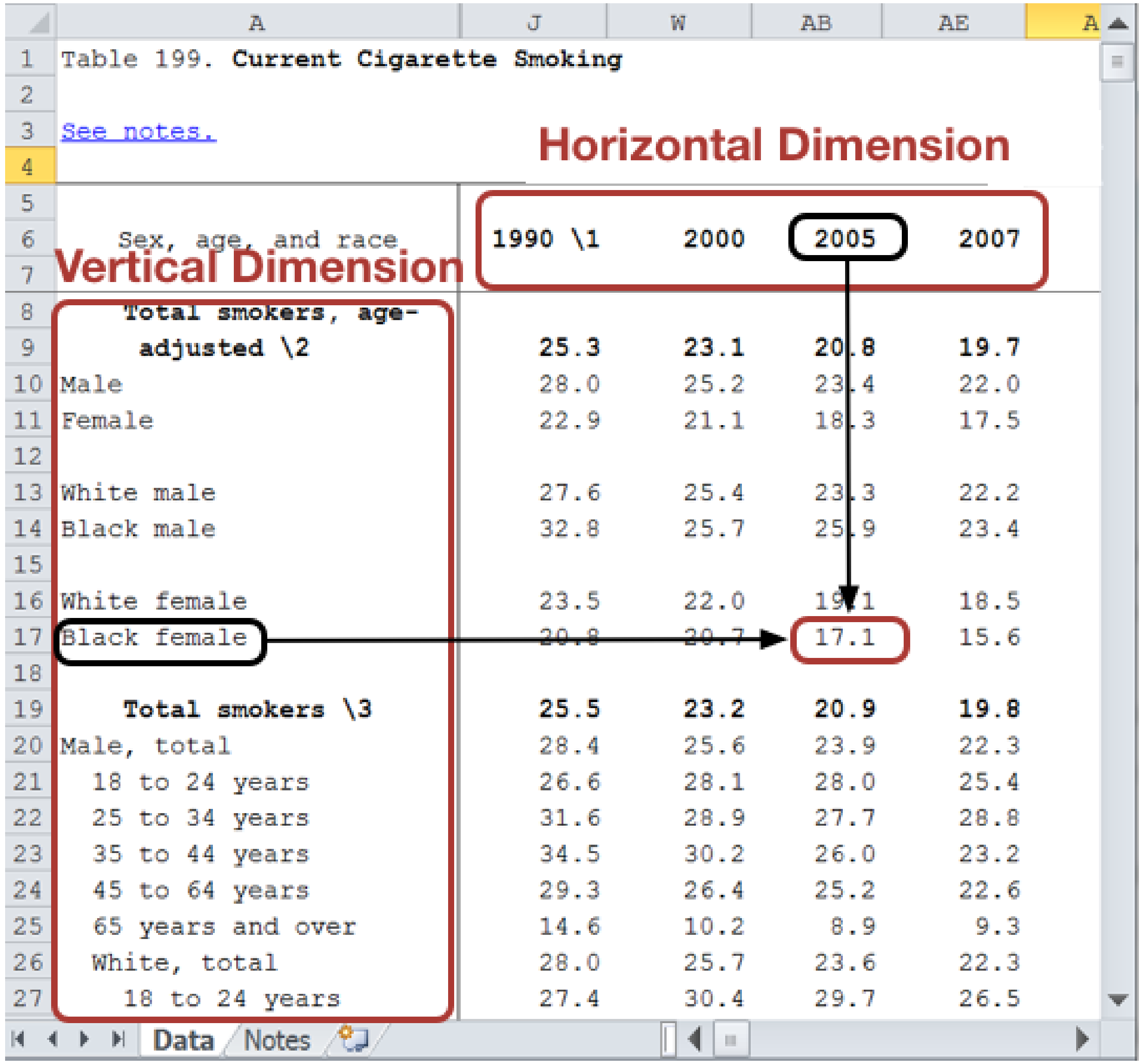
THE DIFFERENCES BETWEEN PIVOT TABLES AND CROSSTABS

Pivot tables and crosstabs are nearly identical in form, and the terms are often used interchangeably. However, pivot tables present some added benefits that regular crosstabs do not.

- Pivot tables allow the user to create additional reports on the spot by easily rearranging, adding, counting, and deleting certain data entries.
- Pivot tables work well with hierarchal organization where data sets can be drilled into to reveal more information. For example, when viewing the total sales at a store by month, you can drill further into the data and see the sales data on individual products for each month. With a basic crosstab, you would have to go back to the program and create a separate crosstab with the information on individual products.
- Pivot tables let the user filter through their data, add or remove custom fields, and change the appearance of their report.

CROSS TAB

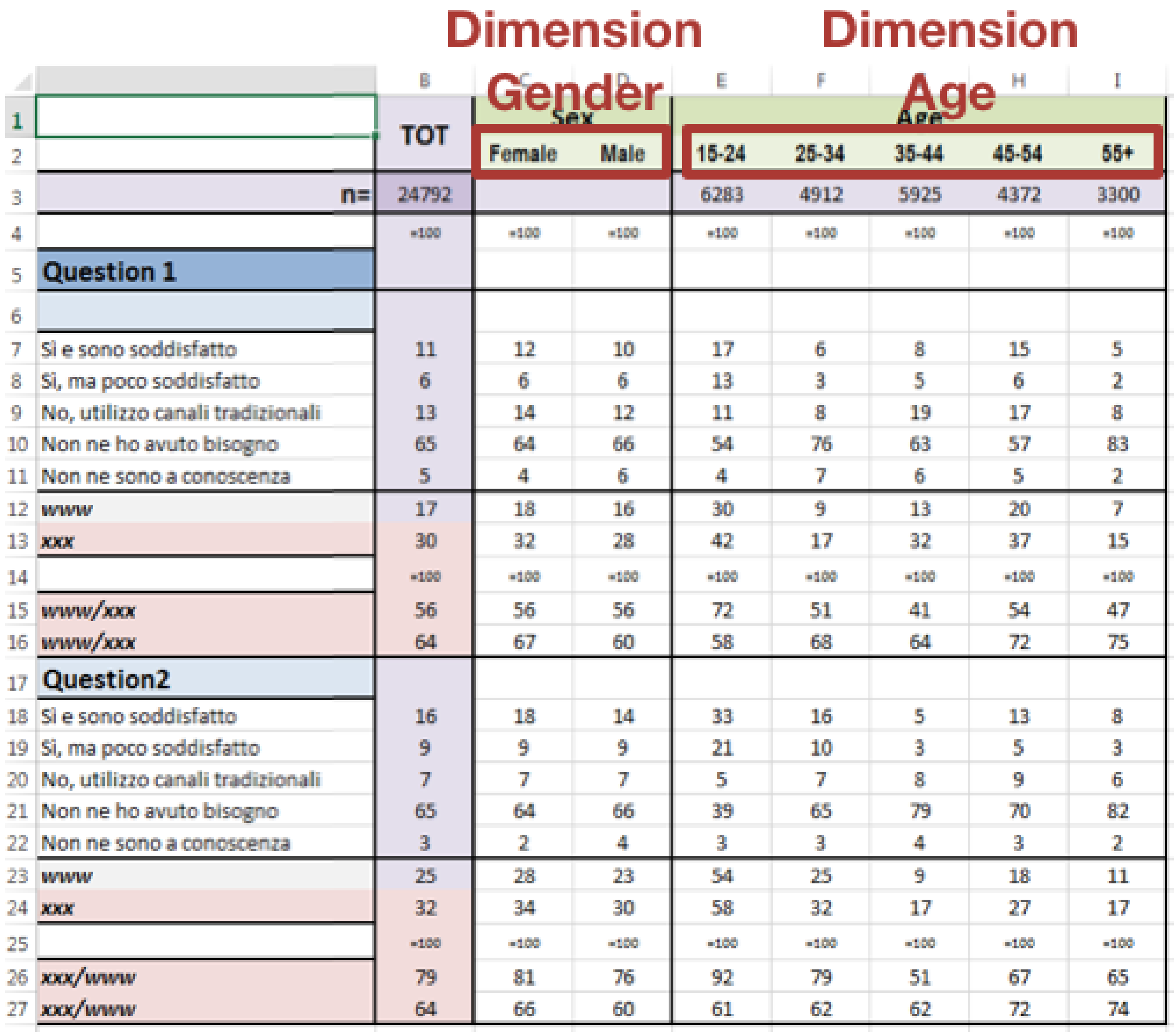
- A cross tab displays data in a row-and-column matrix, with a horizontal dimension and a vertical dimension.
- The values of each dimension often belong to the same class. For example, the values of the horizontal dimension are all "years".
- In a cross tab spreadsheet, each value cell is an intersection of a row field from the vertical dimension and a column field from the horizontal dimension.
- For example, as shown in the cross tab spreadsheet below, the value "17.1" is described by "black female" from the vertical dimension and "2005" from the horizontal dimension.



	A	J	W	AB	AE
1	Table 199. Current Cigarette Smoking				
2					
3	See notes.				
4					
5					
6	Sex, age, and race	1990 \1	2000	2005	2007
7					
8	Total smokers, age-				
9	adjusted \2	25.3	23.1	20.8	19.7
10	Male	28.0	25.2	23.4	22.0
11	Female	22.9	21.1	18.3	17.5
12					
13	White male	27.6	25.4	23.3	22.2
14	Black male	32.8	25.7	25.9	23.4
15					
16	White female	23.5	22.0	19.1	18.5
17	Black female	20.8	20.7	17.1	15.6
18					
19	Total smokers \3	25.5	23.2	20.9	19.8
20	Male, total	28.4	25.6	23.9	22.3
21	18 to 24 years	26.6	28.1	28.0	25.4
22	25 to 34 years	31.6	28.9	27.7	28.8
23	35 to 44 years	34.5	30.2	26.0	23.2
24	45 to 64 years	29.3	26.4	25.2	22.6
25	65 years and over	14.6	10.2	8.9	9.3
26	White, total	28.0	25.7	23.6	22.3
27	18 to 24 years	27.4	30.4	29.7	26.5

PIVOT TABLE

- A pivot spreadsheet is a broader definition similar to cross tab.
- While the cross tab assumes there is only one horizontal dimension and all the values belong to that dimension, pivot assumes there might exist one or more horizontal dimensions.
- For example, the spreadsheet shown below is a pivot spreadsheet where each dimension (dimension gender or dimension age) can be "pivoted" into a new column in the spreadsheet table.


Dimension Gender
Dimension Age

		B	C	D	E	F		H	I
1		TOT	Sex		Age				
2			Female	Male	15-24	25-34	35-44	45-54	55+
3	n=	24792			6283	4912	5925	4372	3300
4		=100	=100	=100	=100	=100	=100	=100	=100
5	Question 1								
6									
7	Sì e sono soddisfatto	11	12	10	17	6	8	15	5
8	Sì, ma poco soddisfatto	6	6	6	13	3	5	6	2
9	No, utilizzo canali tradizionali	13	14	12	11	8	19	17	8
10	Non ne ho avuto bisogno	65	64	66	54	76	63	57	83
11	Non ne sono a conoscenza	5	4	6	4	7	6	5	2
12	www	17	18	16	30	9	13	20	7
13	xxx	30	32	28	42	17	32	37	15
14		=100	=100	=100	=100	=100	=100	=100	=100
15	www/xxx	56	56	56	72	51	41	54	47
16	www/xxx	64	67	60	58	68	64	72	75
17	Question2								
18	Sì e sono soddisfatto	16	18	14	33	16	5	13	8
19	Sì, ma poco soddisfatto	9	9	9	21	10	3	5	3
20	No, utilizzo canali tradizionali	7	7	7	5	7	8	9	6
21	Non ne ho avuto bisogno	65	64	66	39	65	79	70	82
22	Non ne sono a conoscenza	3	2	4	3	3	4	3	2
23	www	25	28	23	54	25	9	18	11
24	xxx	32	34	30	58	32	17	27	17
25		=100	=100	=100	=100	=100	=100	=100	=100
26	xxx/www	79	81	76	92	79	51	67	65
27	xxx/www	64	66	60	61	62	62	72	74

	site	Product_Category	Product	Sales
0	walmart	Kitchen	Oven	2000
1	amazon	Home-Decor	Sofa-set	3000
2	alibaba	Gardening	digging spade	4000
3	flipkart	Health	fitness band	5000
4	alibaba	Beauty	sunscreen	6000
5	flipkart	Garments	pyjamas	9000
6	walmart	Gardening	digging spade	3000
7	amazon	Health	fitness band	2500
8	alibaba	Beauty	sunscreen	1020
9	flipkart	Garments	pyjamas	950

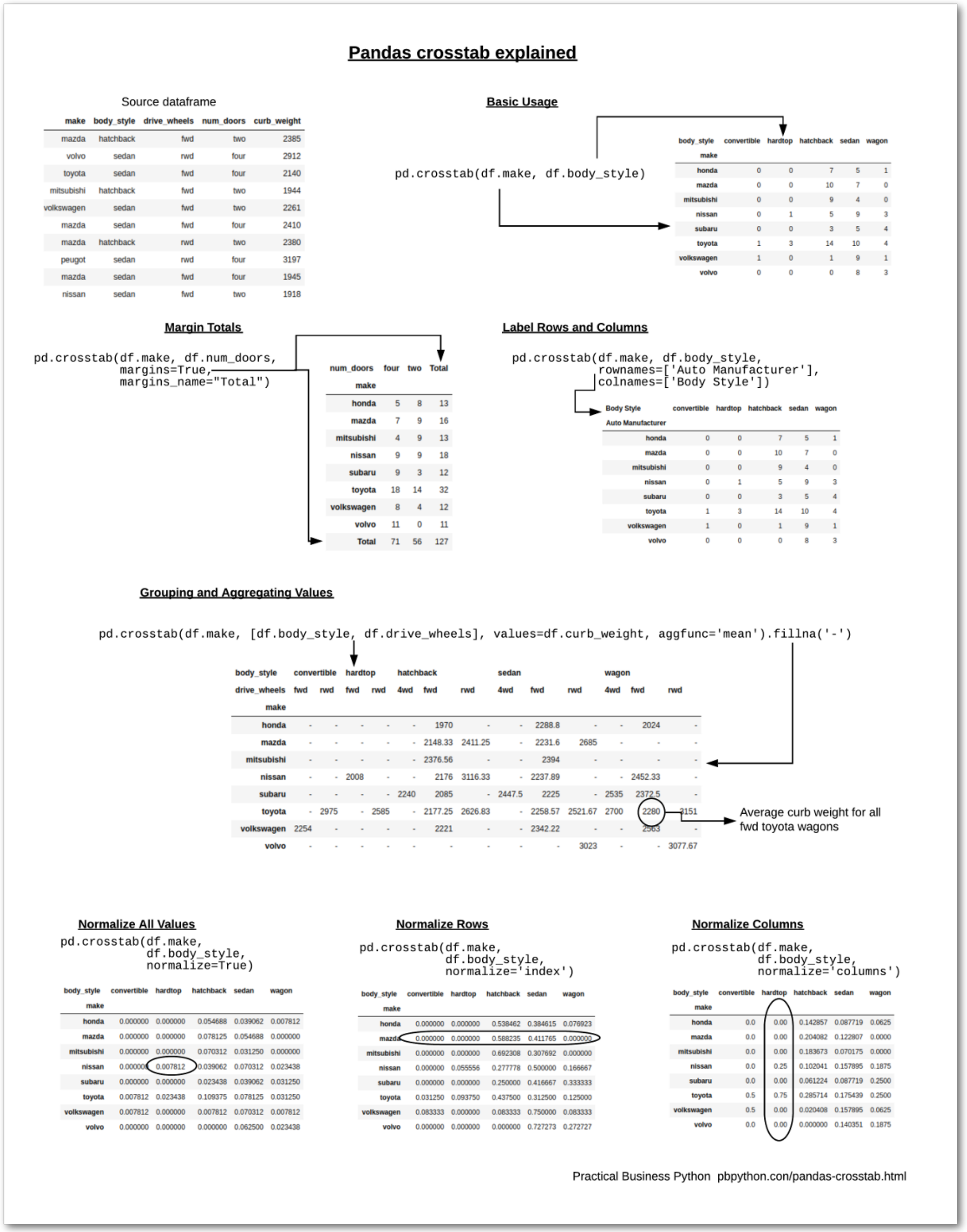
		Sales			
	site	alibaba	amazon	flipkart	walmart
Product_Category	Product				
Beauty	sunscreen	3510.0	NaN	NaN	NaN
Gardening	digging spade	4000.0	NaN	NaN	3000.0
Garments	pyjamas	NaN	NaN	4975.0	NaN
Health	fitness band	NaN	2500.0	5000.0	NaN
Home-Decor	Sofa-set	NaN	3000.0	NaN	NaN
Kitchen	Oven	NaN	NaN	NaN	2000.0

PIVOT TABLE

	site	alibaba	amazon	flipkart	walmart
Product_Category	Product				
Beauty	sunscreen	2	0	0	0
Gardening	digging spade	1	0	0	1
Garments	pyjamas	0	0	2	0
Health	fitness band	0	1	1	0
Home-Decor	Sofa-set	0	1	0	0
Kitchen	Oven	0	0	0	1

CROSS TABULATION

CROSS TABULATION



Pandas crosstab explained

Source dataframe

make	body_style	drive_wheels	num_doors	curb_weight
mazda	hatchback	fwd	two	2385
volvo	sedan	rwd	four	2912
toyota	sedan	fwd	four	2140
mitsubishi	hatchback	fwd	two	1944
volkswagen	sedan	fwd	two	2261
mazda	sedan	fwd	four	2410
mazda	hatchback	rwd	two	2380
peugot	sedan	rwd	four	3197
mazda	sedan	fwd	four	1945
nissan	sedan	fwd	two	1918

Basic Usage

```
pd.crosstab(df.make, df.body_style)
```

body_style	convertible	hardtop	hatchback	sedan	wagon
make					
honda	0	0	7	5	1
mazda	0	0	10	7	0
mitsubishi	0	0	9	4	0
nissan	0	1	5	9	3
subaru	0	0	3	5	4
toyota	1	3	14	10	4
volkswagen	1	0	1	9	1
volvo	0	0	0	8	3

Margin Totals

```
pd.crosstab(df.make, df.num_doors,
            margins=True,
            margins_name="Total")
```

num_doors	four	two	Total
make			
honda	5	8	13
mazda	7	9	16
mitsubishi	4	9	13
nissan	9	9	18
subaru	9	3	12
toyota	18	14	32
volkswagen	8	4	12
volvo	11	0	11
Total	71	56	127

Label Rows and Columns

```
pd.crosstab(df.make, df.body_style,
            rownames=['Auto Manufacturer'],
            colnames=['Body Style'])
```

Body Style	convertible	hardtop	hatchback	sedan	wagon
Auto Manufacturer					
honda	0	0	7	5	1
mazda	0	0	10	7	0
mitsubishi	0	0	9	4	0
nissan	0	1	5	9	3
subaru	0	0	3	5	4
toyota	1	3	14	10	4
volkswagen	1	0	1	9	1
volvo	0	0	0	8	3

Grouping and Aggregating Values

```
pd.crosstab(df.make, [df.body_style, df.drive_wheels], values=df.curb_weight, aggfunc='mean').fillna('-')
```

body_style	convertible		hardtop		hatchback			sedan			wagon		
drive_wheels	fwd	rwd	fwd	rwd	4wd	fwd	rwd	4wd	fwd	rwd	4wd	fwd	rwd
make													
honda	-	-	-	-	-	1970	-	-	2288.8	-	-	2024	-
mazda	-	-	-	-	-	2148.33	2411.25	-	2231.6	2685	-	-	-
mitsubishi	-	-	-	-	-	2376.56	-	-	2394	-	-	-	-
nissan	-	-	2008	-	-	2176	3116.33	-	2237.89	-	-	2452.33	-
subaru	-	-	-	-	2240	2085	-	2447.5	2225	-	2535	2372.5	-
toyota	-	2975	-	2585	-	2177.25	2626.83	-	2258.57	2521.67	2700	2280	3151
volkswagen	2254	-	-	-	-	2221	-	-	2342.22	-	-	2563	-
volvo	-	-	-	-	-	-	-	-	-	3023	-	-	3077.67

Average curb weight for all fwd toyota wagons

Normalize All Values

```
pd.crosstab(df.make,
            df.body_style,
            normalize=True)
```

body_style	convertible	hardtop	hatchback	sedan	wagon
make					
honda	0.000000	0.000000	0.054688	0.039062	0.007812
mazda	0.000000	0.000000	0.078125	0.054688	0.000000
mitsubishi	0.000000	0.000000	0.070312	0.031250	0.000000
nissan	0.000000	0.007812	0.039062	0.070312	0.023438
subaru	0.000000	0.000000	0.023438	0.039062	0.031250
toyota	0.007812	0.023438	0.109375	0.078125	0.031250
volkswagen	0.007812	0.000000	0.007812	0.070312	0.007812
volvo	0.000000	0.000000	0.000000	0.062500	0.023438

Normalize Rows

```
pd.crosstab(df.make,
            df.body_style,
            normalize='index')
```

body_style	convertible	hardtop	hatchback	sedan	wagon
make					
honda	0.000000	0.000000	0.538462	0.384615	0.076923
mazda	0.000000	0.000000	0.588235	0.411765	0.000000
mitsubishi	0.000000	0.000000	0.692308	0.307692	0.000000
nissan	0.000000	0.055556	0.277778	0.500000	0.166667
subaru	0.000000	0.000000	0.250000	0.416667	0.333333
toyota	0.031250	0.093750	0.437500	0.312500	0.125000
volkswagen	0.083333	0.000000	0.083333	0.750000	0.083333
volvo	0.000000	0.000000	0.000000	0.727273	0.272727

Normalize Columns

```
pd.crosstab(df.make,
            df.body_style,
            normalize='columns')
```

body_style	convertible	hardtop	hatchback	sedan	wagon
make					
honda	0.0	0.00	0.142857	0.087719	0.0625
mazda	0.0	0.00	0.204082	0.122807	0.0000
mitsubishi	0.0	0.00	0.183673	0.070175	0.0000
nissan	0.0	0.25	0.102041	0.157895	0.1875
subaru	0.0	0.00	0.061224	0.087719	0.2500
toyota	0.5	0.75	0.285714	0.175439	0.2500
volkswagen	0.5	0.00	0.020408	0.157895	0.0625
volvo	0.0	0.00	0.000000	0.140351	0.1875

Practical Business Python pbpython.con/pandas-crosstab.html

Data transformation allows organizations to turn data from various locations and formats into actionable insights. It accomplishes this by streamlining the processes which refine, standardize, and consolidate these many types of data.

THANK YOU

khairulbazli@ump.edu.my