DATA VISUALIZATION

LAB 3

HOW TO PICK THE RIGHT CHART?

First, we need to determine the goal or the purpose of this visualization.

The goal of visualization can be categorized into:

ICCOR

Inform, Compare, Change, Organize, Reveal relationships

Inform: convey a single important message or data point that doesn’t require much context to understand

Compare: show similarities or differences among values or parts of a whole

Change: visualize trends over time or space

Organize: show groups, patterns, rank, or order

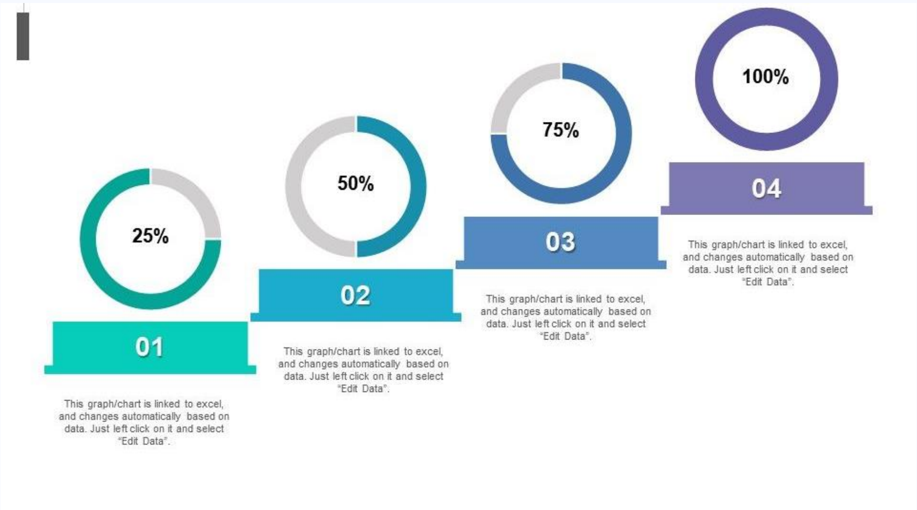
Reveal Relationships: show correlations among variables or values

1. INFORM

1. SINGLE BIG NUMBER

43%
SINGLE NUMBER

2. DONUT CHART(SIMPLE PROPORTION)



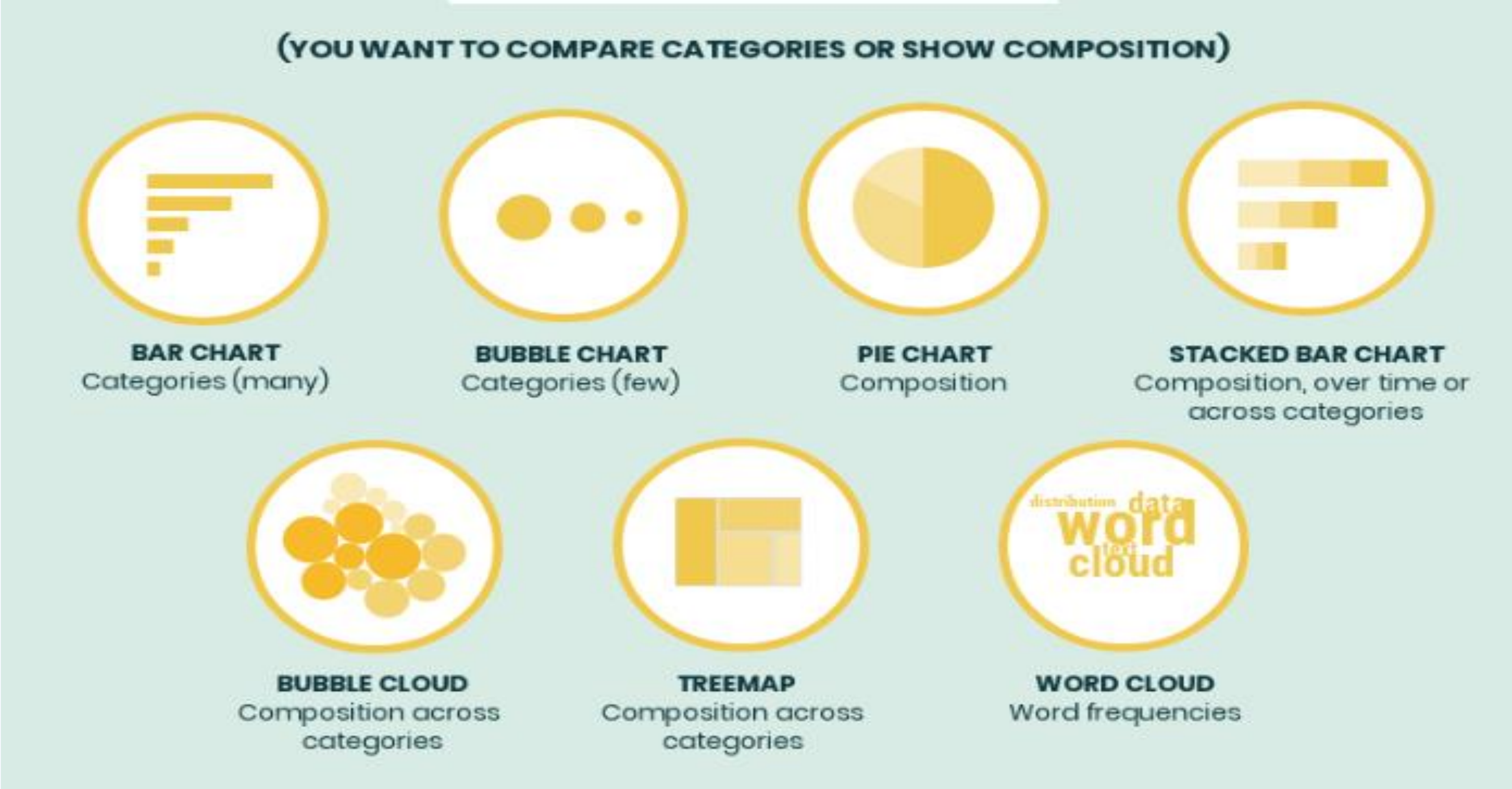
This graph/chart is linked to excel, and changes automatically based on data. Just left click on it and select "Edit Data".

This graph/chart is linked to excel, and changes automatically based on data. Just left click on it and select "Edit Data".

This graph/chart is linked to excel, and changes automatically based on data. Just left click on it and select "Edit Data".

This graph/chart is linked to excel, and changes automatically based on data. Just left click on it and select "Edit Data".

2. COMPARE



(YOU WANT TO COMPARE CATEGORIES OR SHOW COMPOSITION)

BAR CHART
Categories (many)

BUBBLE CHART
Categories (few)

PIE CHART
Composition

STACKED BAR CHART
Composition, over time or across categories

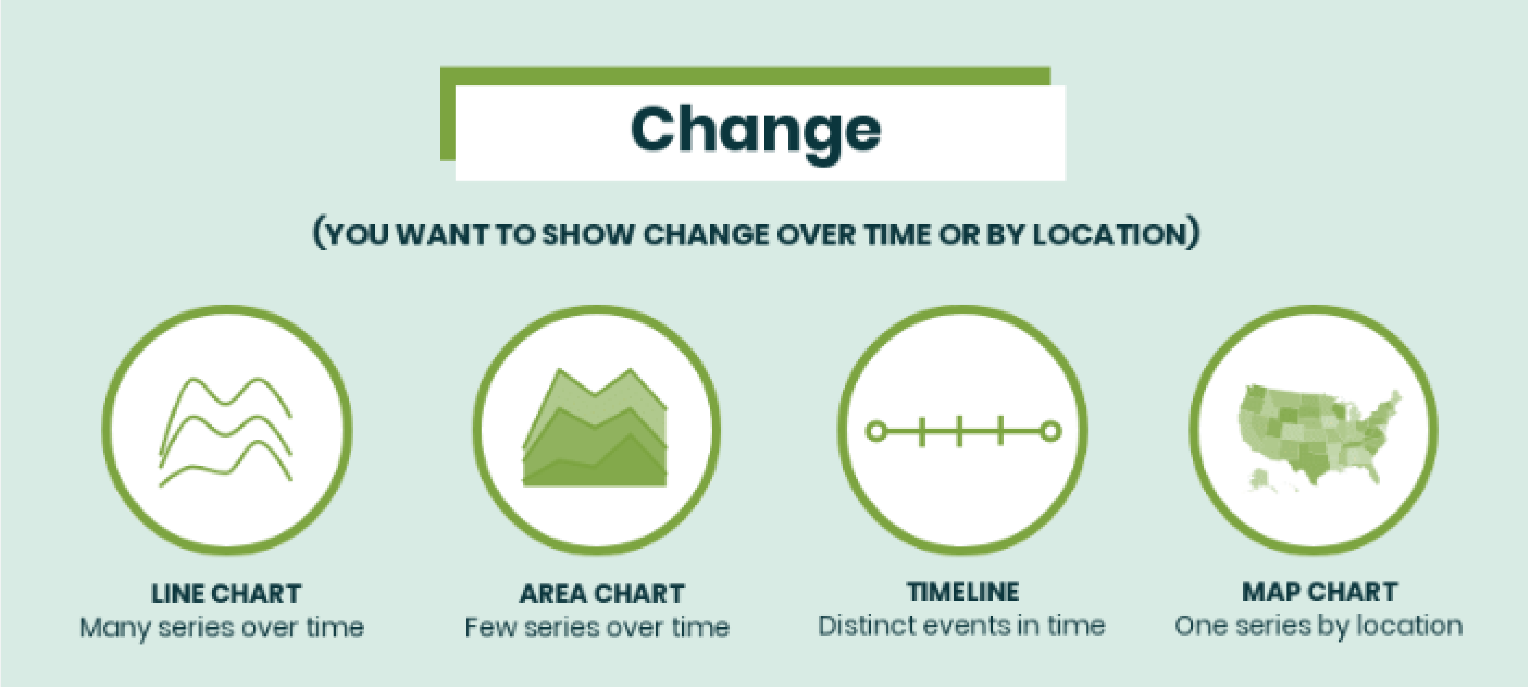
BUBBLE CLOUD
Composition across categories

TREEMAP
Composition across categories

WORD CLOUD
Word frequencies

The Chart	When to use it
Bar chart(vertical)	Bar graphs are used to compare things between different groups or to track changes over time. However, when trying to measure change over time, bar graphs are best when the changes are larger.
Bar chart(Horizontal)	A horizontal bar chart is a great option for **long category names**
Stacked bar chart	best used when showing comparisons between categories. Typically, the bars are proportional to the values they represent and can be plotted either horizontally or vertically.
Pie chart	Pie charts are best to use when you are trying to compare parts of a whole. They do not show changes over time.(only used with few categories)

3. CHANGE



Change

(YOU WANT TO SHOW CHANGE OVER TIME OR BY LOCATION)

LINE CHART
Many series over time

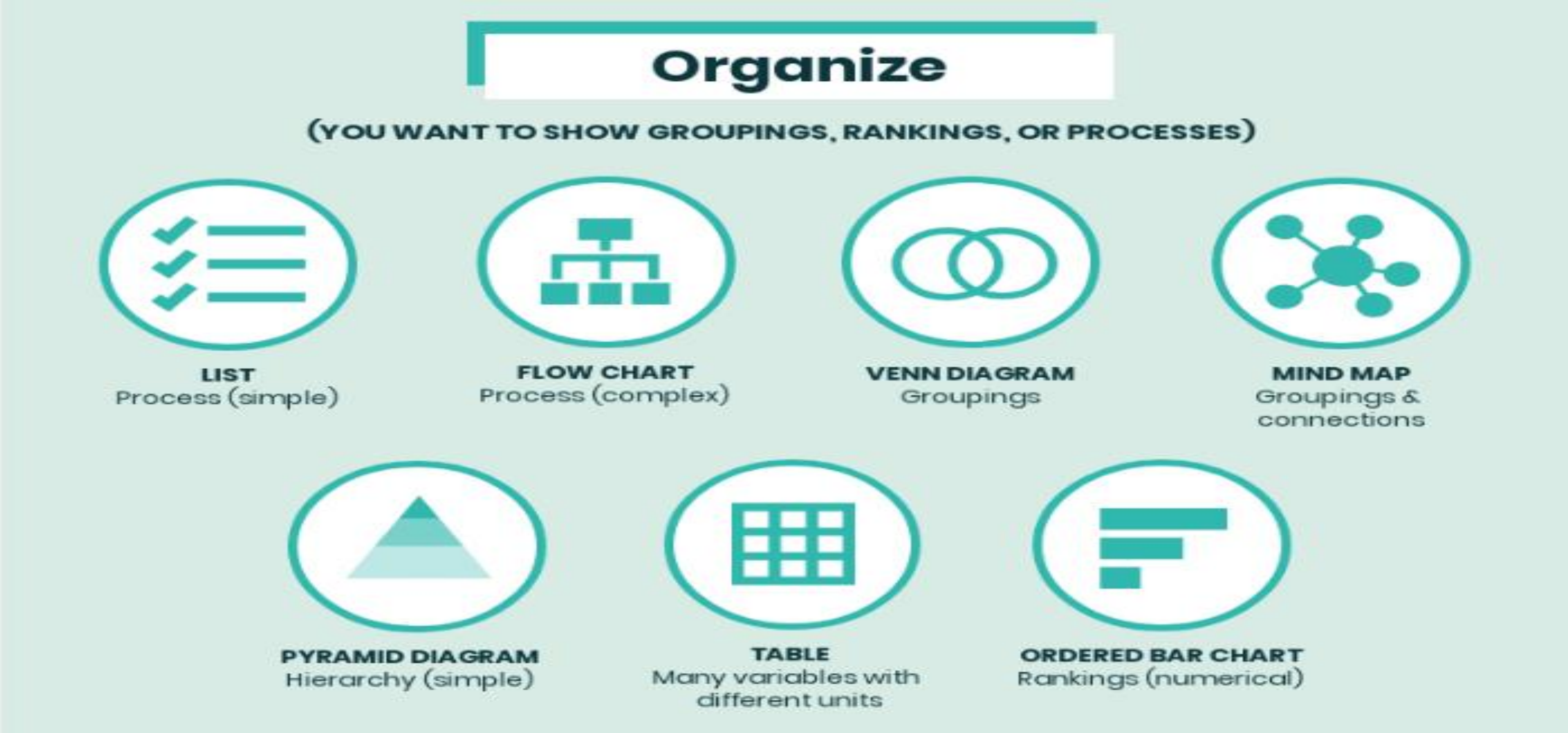
AREA CHART
Few series over time

TIMELINE
Distinct events in time

MAP CHART
One series by location

The Chart	When to use it
Line chart	Line graphs are used to track changes over short and long periods of time. When smaller changes exist, line graphs are better to use than bar graphs. Line graphs can also be used to compare changes over the same period of time for more than one group.
Area chart	Area graphs are very similar to line graphs. They can be used to track changes over time for one or more groups. Area graphs are good to use when you are tracking the changes in two or more related groups that make up one whole category (for example public and private groups), no more than 4 groups.
Map chart	help visualize geographically based data, such as population density, election results, or economic indicators

4.ORGANIZE



Organize

(YOU WANT TO SHOW GROUPINGS, RANKINGS, OR PROCESSES)

LIST
Process (simple)

FLOW CHART
Process (complex)

VENN DIAGRAM
Groupings

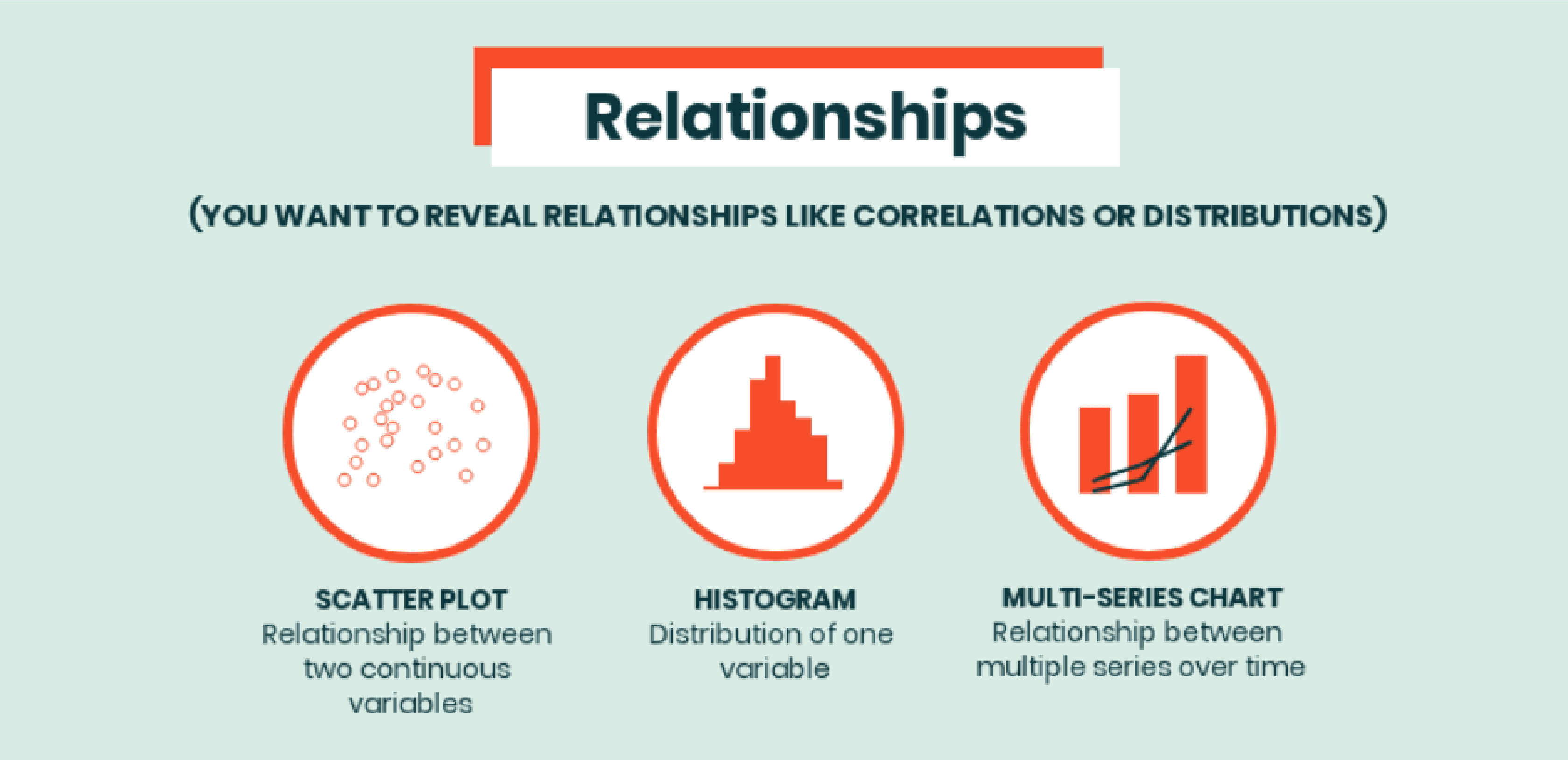
MIND MAP
Groupings & connections

PYRAMID DIAGRAM
Hierarchy (simple)

TABLE
Many variables with different units

ORDERED BAR CHART
Rankings (numerical)

5.REVEAL RELATIONSHIPS



Relationships

(YOU WANT TO REVEAL RELATIONSHIPS LIKE CORRELATIONS OR DISTRIBUTIONS)

SCATTER PLOT
Relationship between two continuous variables

HISTOGRAM
Distribution of one variable

MULTI-SERIES CHART
Relationship between multiple series over time

The Chart	When to use it
Scatter plot	observe and show relationships between two numeric variables. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole. Identification of correlational relationships are common with scatter plots.
Histogram	summarize discrete or continuous data that are measured on an interval scale. It is often used to illustrate the major features of the distribution of the data in a convenient form.
Multi-series chart	allow to plot data for multiple datasets. For example, you can plot the revenue collected each month for the last two years using a multi-series chart.

IMPORTING PACKAGES TO BE USED

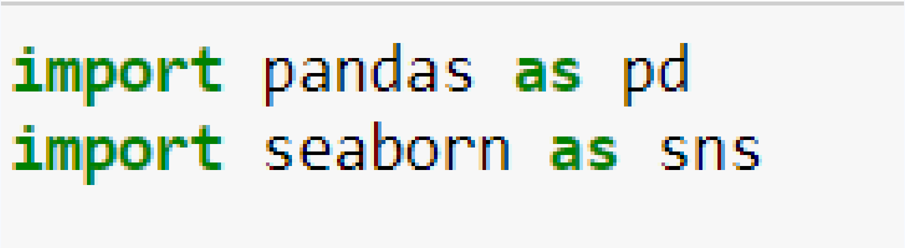
1. pandas (pd)

2. numpy (np)

2. Matplotlib, Matplotib.pyplot (plt)

3. seaborn (sns)

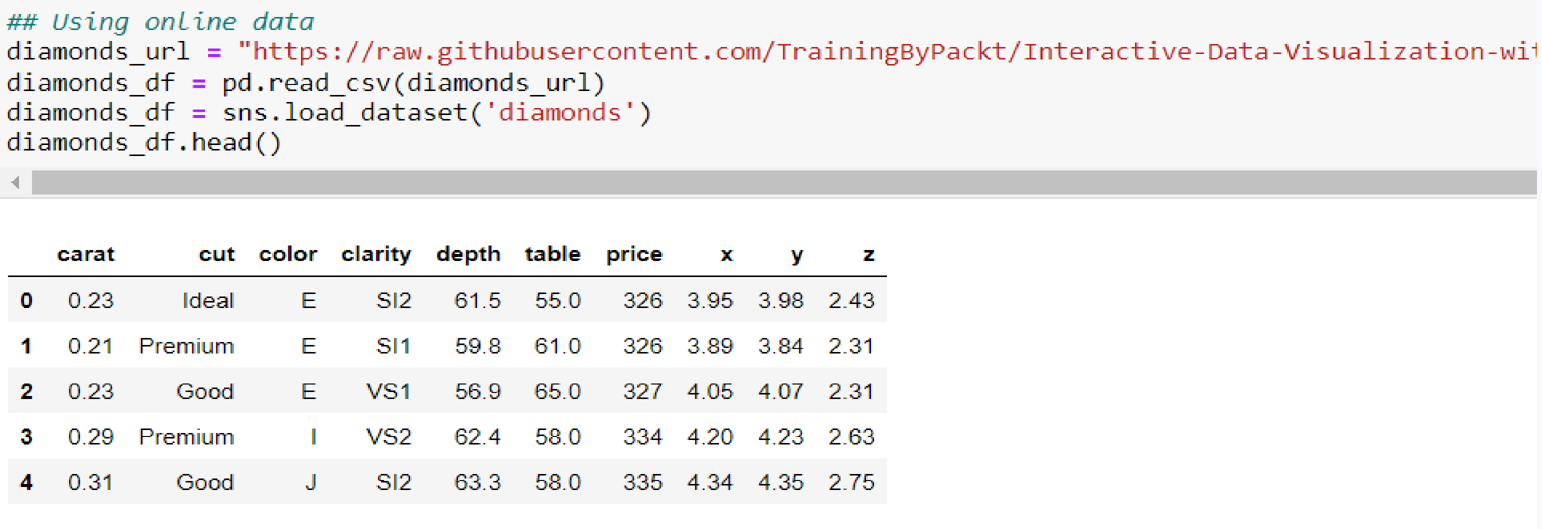
4. plotly.express (px)



```
import pandas as pd
import seaborn as sns
```

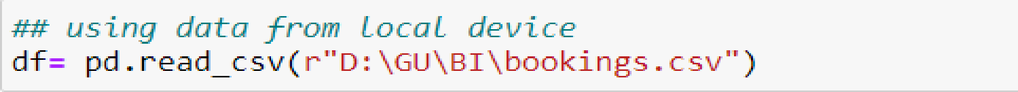
IMPORT, OBSERVE AND DESCRIBE THE DATA

The data can be imported whether from your local device or a hyper link of an online dataset, in most cases we use pandas to read the data.



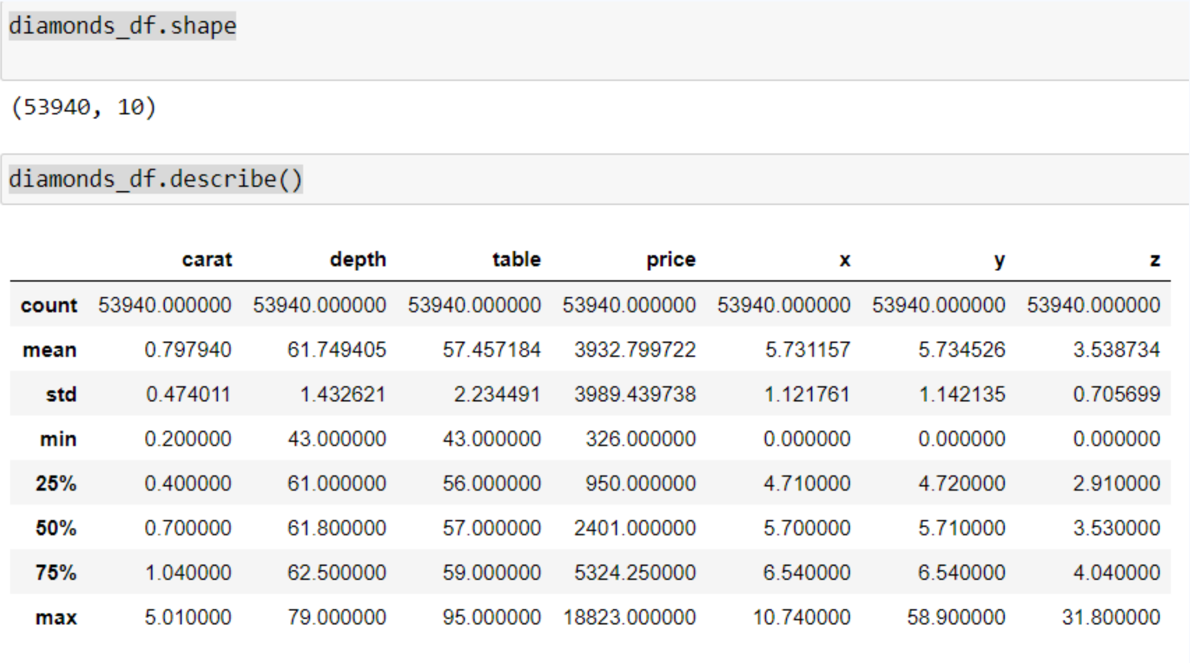
```
## Using online data
diamonds_url = "https://raw.githubusercontent.com/TrainingByPackt/Interactive-Data-Visualization-wi
diamonds_df = pd.read_csv(diamonds_url)
diamonds_df = sns.load_dataset('diamonds')
diamonds_df.head()
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75



```
## using data from local device
df= pd.read_csv(r"D:\GU\BI\bookings.csv")
```

IMPORT, OBSERVE AND DESCRIBE THE DATA

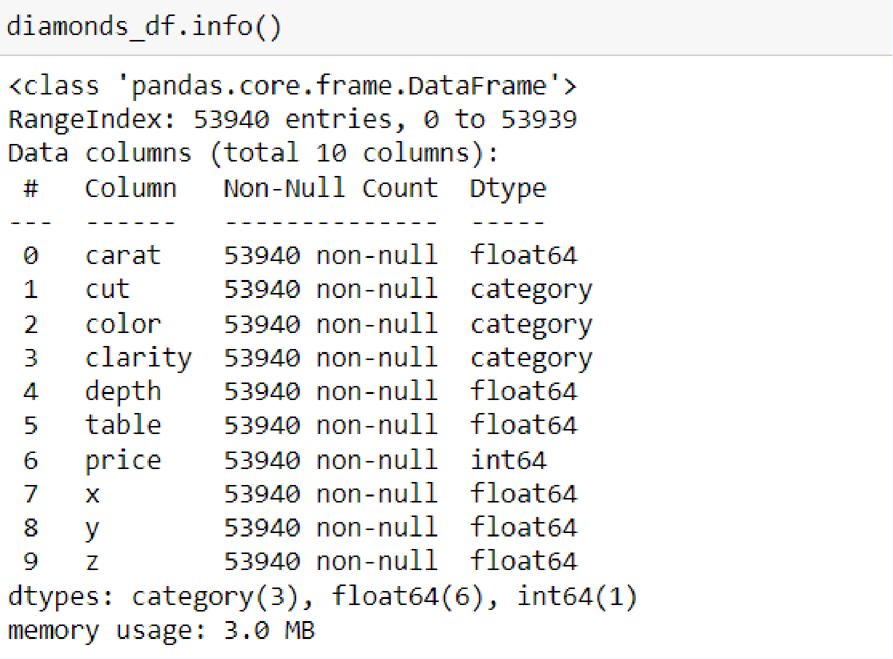


```
diamonds_df.shape
```

```
(53940, 10)
```

```
diamonds_df.describe()
```

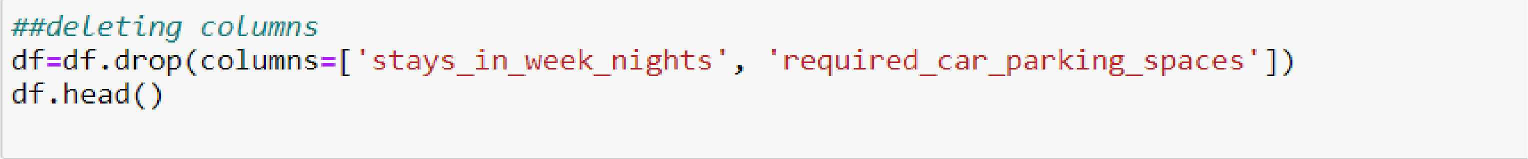
	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000



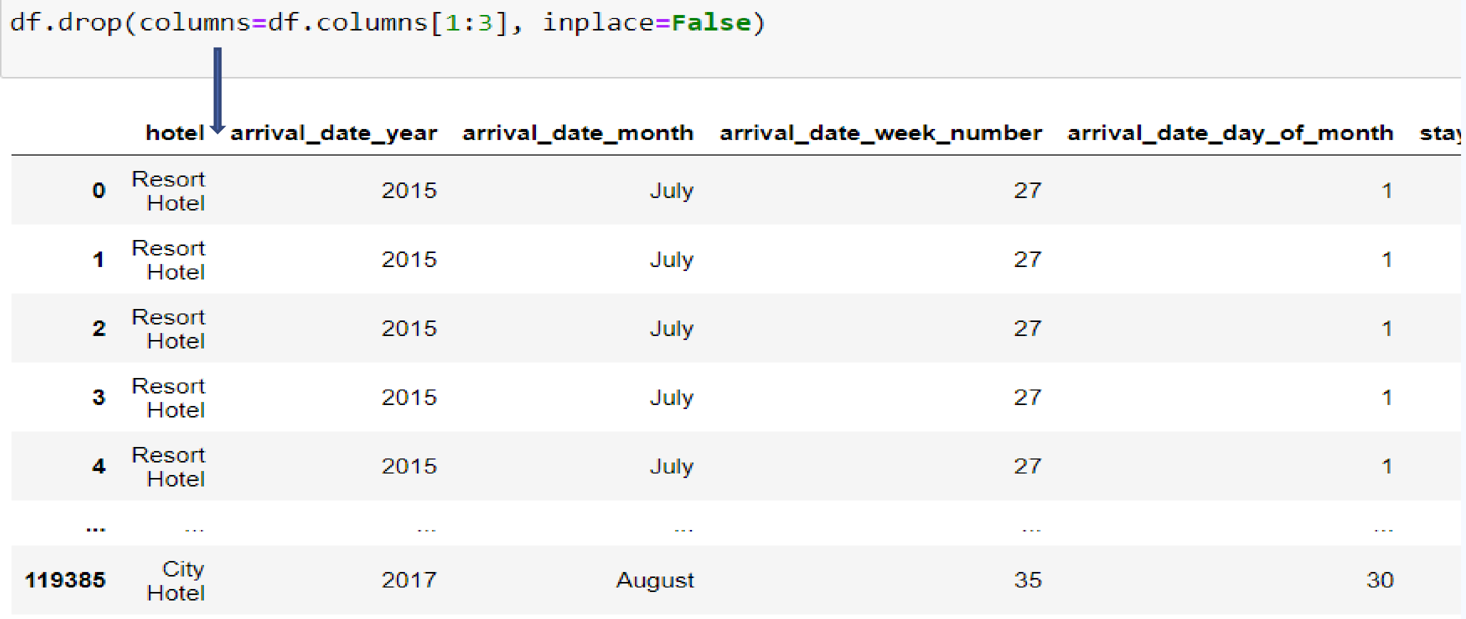
```
diamonds_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    53940 non-null  float64
 1   cut      53940 non-null  category
 2   color    53940 non-null  category
 3   clarity  53940 non-null  category
 4   depth    53940 non-null  float64
 5   table    53940 non-null  float64
 6   price    53940 non-null  int64
 7   x        53940 non-null  float64
 8   y        53940 non-null  float64
 9   z        53940 non-null  float64
dtypes: category(3), float64(6), int64(1)
memory usage: 3.0 MB
```

DELETING



```
##deleting columns
df=df.drop(columns=['stays_in_week_nights', 'required_car_parking_spaces'])
df.head()
```



```
df.drop(columns=df.columns[1:3], inplace=False)
```

	hotel	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stay
0	Resort Hotel	2015	July	27	1	
1	Resort Hotel	2015	July	27	1	
2	Resort Hotel	2015	July	27	1	
3	Resort Hotel	2015	July	27	1	
4	Resort Hotel	2015	July	27	1	
...	...	...	...	...	...	
119385	City Hotel	2017	August	35	30	

DELETING MISSING VALUES

df.dropna() ... this will drop all rows with null values

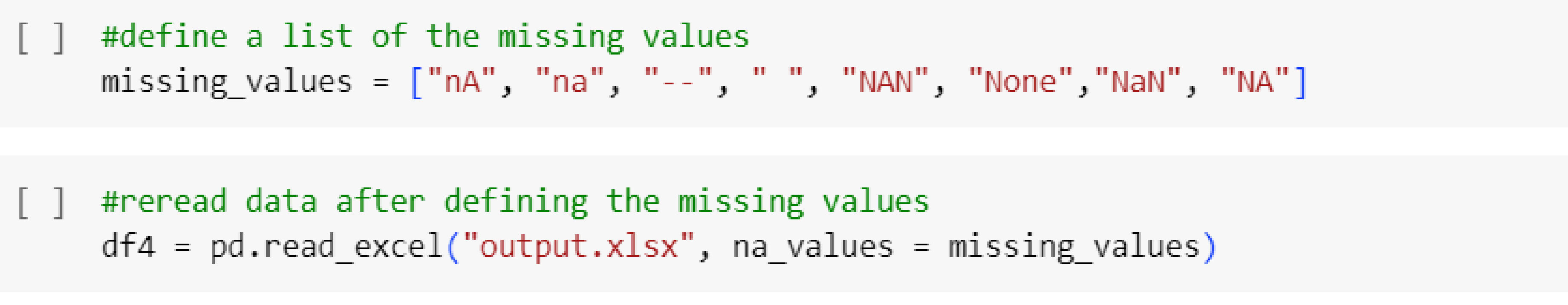
df.dropna(axis=1) ... drops all columns that contain null values

df.dropna(axis=1, how='all') ... drops columns that all of their values are null values

dropna(axis=1, how='any') .... drops all columns with any NA values

df.dropna(axis=1, thresh=) ... drops columns that have less than n not null values

DEALING WITH MISSING VALUES



```
[ ]  #define a list of the missing values
     missing_values = ["nA", "na", "--", " ", "NAN", "None","NaN", "NA"]
```

```
[ ]  #reread data after defining the missing values
     df4 = pd.read_excel("output.xlsx", na_values = missing_values)
```