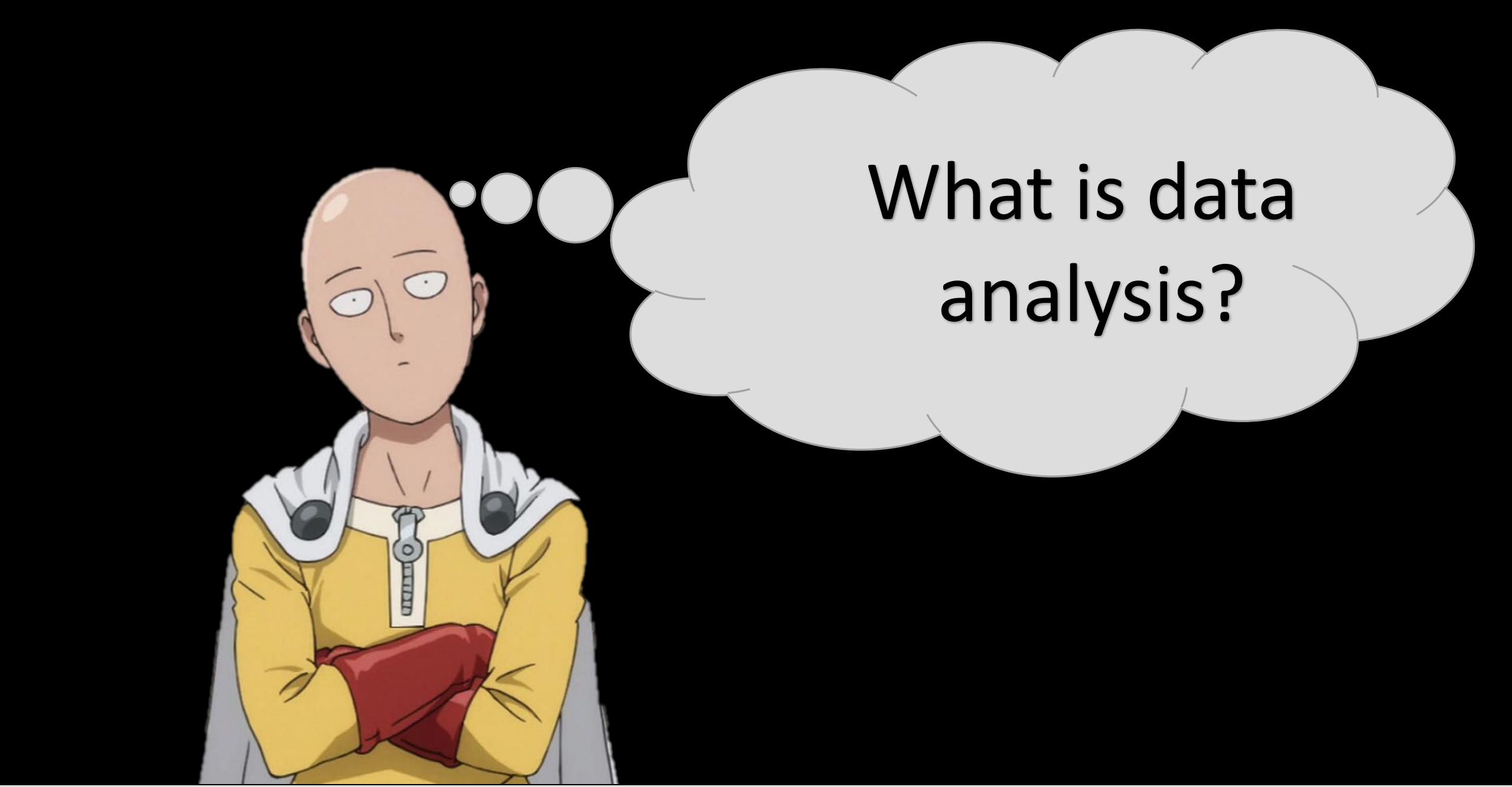
DATA ANALYSIS WITH PYTHON

OUTLINE

1. Data Analysis Definition
2. Reason Analyze Data with Python
3. How Analyze Data with Python
4. How visualize data with Python


What is data analysis?

Data analysis is defined as a

process of cleaning, transforming, and modeling data to discover useful information for decision making. The purpose of data analysis is to extract useful information from data and taking the decision based upon the data analysis

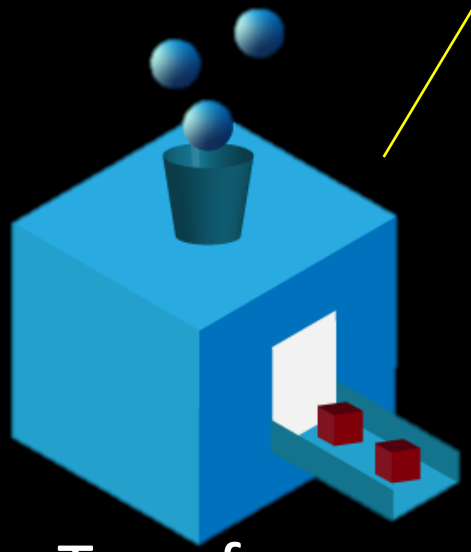
Why we use Python to analyze data?


1. Code would be easy for humans to read
2. Open source and has large library
3. Large community and easy to collaborate
4. Can corelate with others tools
5. Can to make visualization & graphics

How to analyze data with Python?

Cleaning

Pandas



Transform

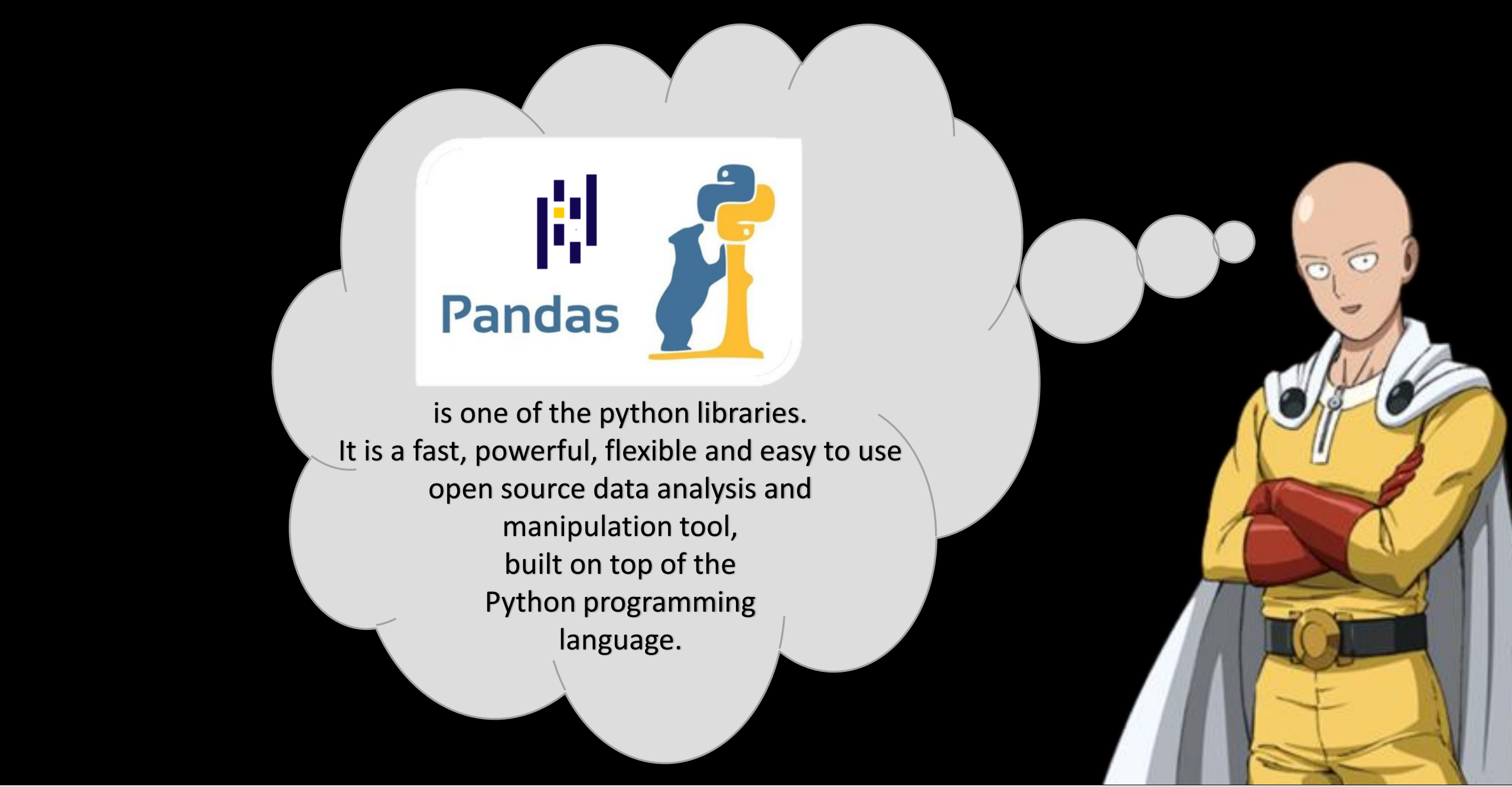
Visualization

Matplotlib, Seaborn,
Folium, plotly

What is pandas?

Is this pandas?

Or is this?


Pandas
is one of the python libraries.
It is a fast, powerful, flexible and easy to use
open source data analysis and
manipulation tool,
built on top of the
Python programming
language.

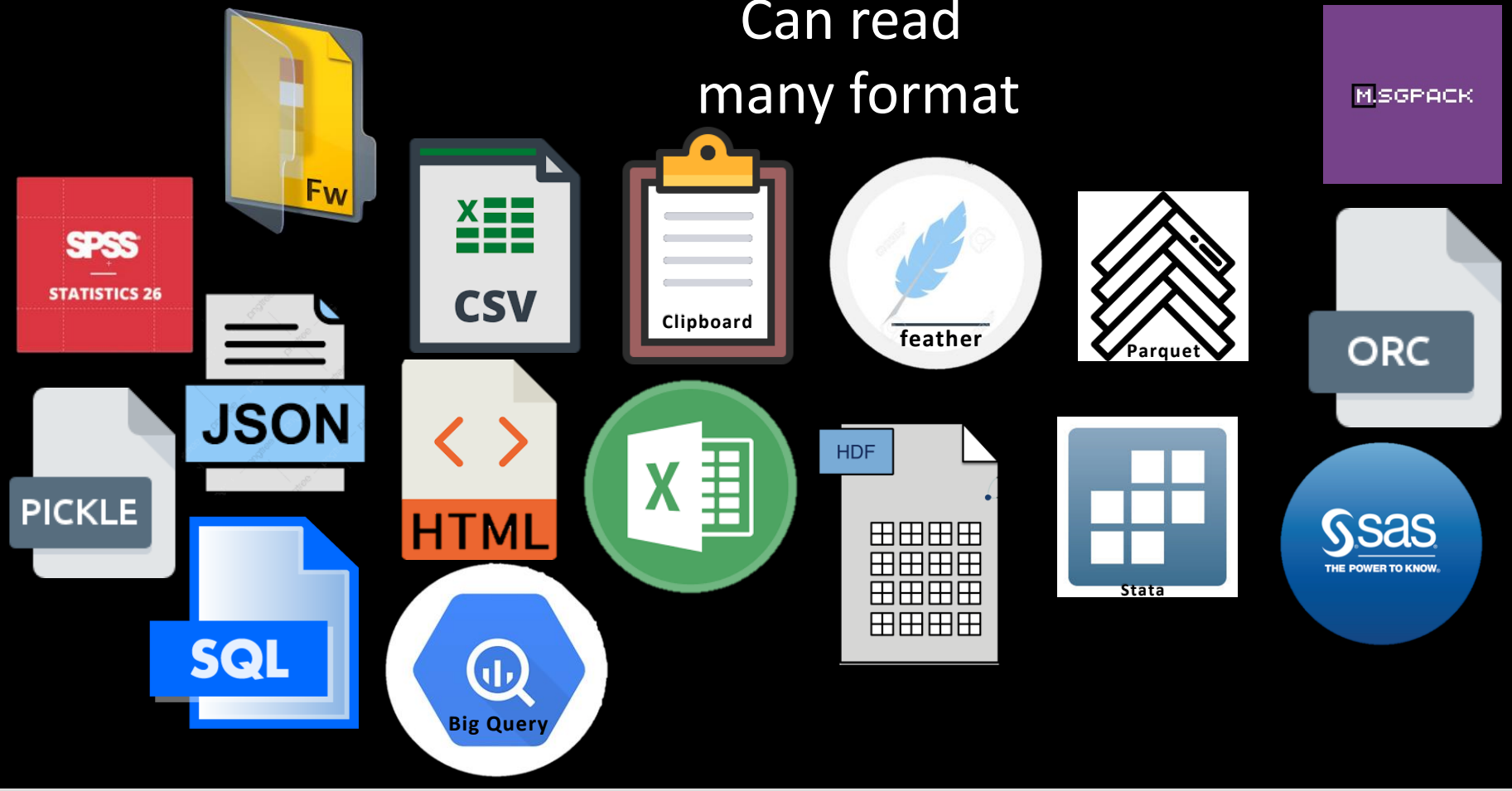

Pandas
Why Pandas is so powerful?

Because

1. Built on top of Numpy package, hikes its performance.
2. Customized indexing for Data Frames
3. Tools for loading data into in memory data objects from different file formats
4. Data alignment and effective handling of missing data
5. Reshaping of the dataset is possible
6. Slicing, indexing, and sub setting of large data sets
7. Deletion or insertion of columns
8. Group by data for aggregation and transformations
9. High-performance merging and joining of data
10. Time series functionality

Pandas

Can read
many format



How to read data in pandas?

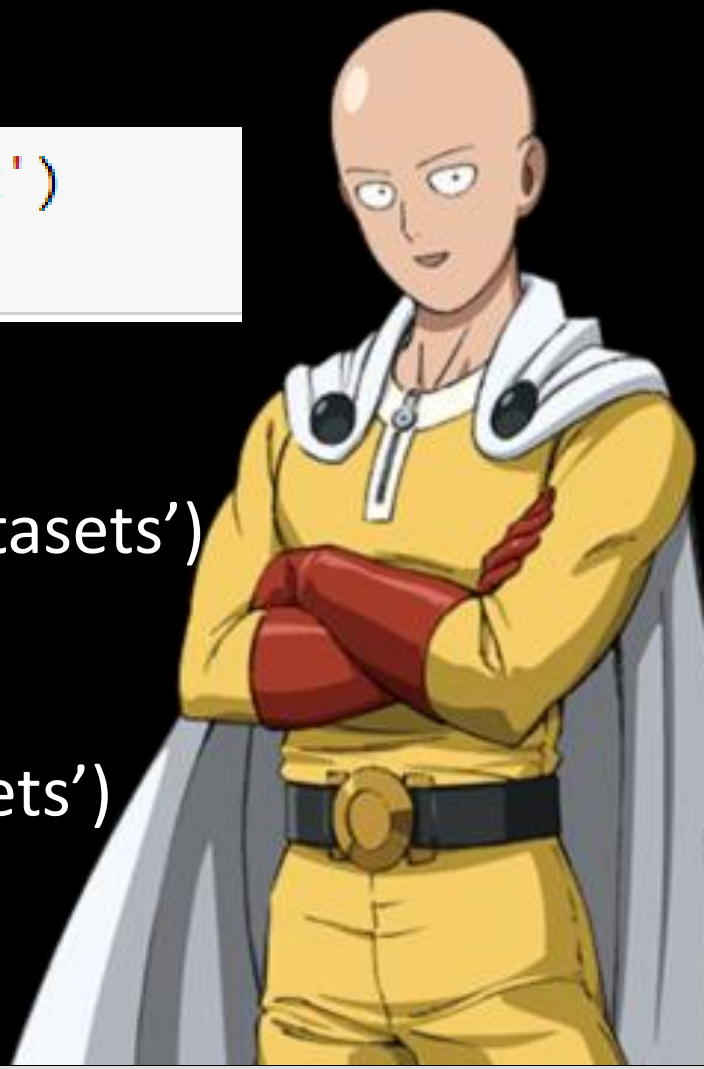
```
dc_ritz = pd.read_excel('data_dictionary_Ritz_jager.xlsx')
ritz_jager = pd.read_csv('Ritz_Jager_Data.csv')
```

To read excel

pd.read_excel(‘ this your path data sets and your name datasets’)

To read csv

pd.read_csv(‘ this your path data sets and your name datasets’)



In Pandas there is dataframe

Index

Columns/feature

Rows/records

	company	agent	country_origin	children
0	NaN	NaN	PRT	0.0
1	NaN	NaN	PRT	0.0
2	NaN	NaN	GBR	0.0
3	NaN	304.0	GBR	0.0
4	NaN	240.0	GBR	0.0
...	...	...	...	...
119385	NaN	394.0	BEL	0.0
119386	NaN	9.0	FRA	0.0
119387	NaN	9.0	DEU	0.0
119388	NaN	89.0	GBR	0.0
119389	NaN	9.0	DEU	0.0

119390 rows × 4 columns

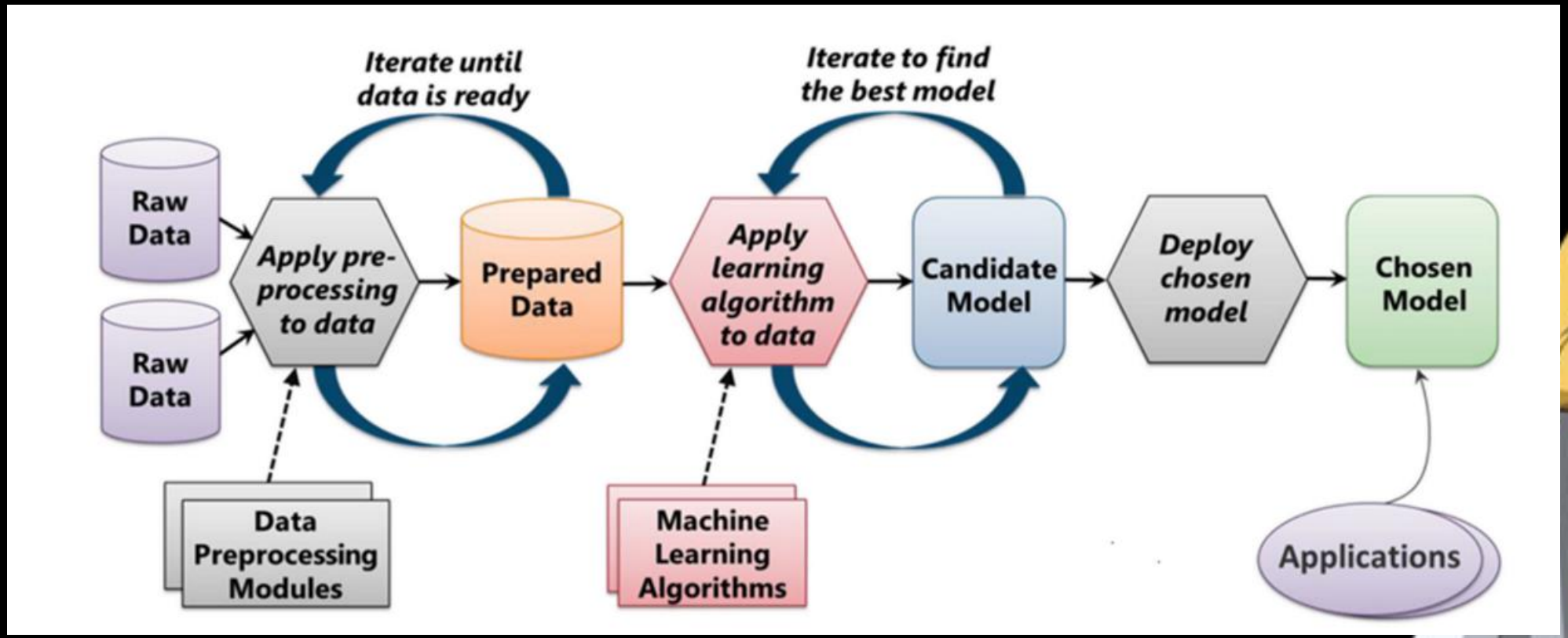
dataframe

	agent
0	NaN
1	NaN
2	NaN
3	304.0
4	240.0
...	...
119385	394.0
119386	9.0
119387	9.0
119388	89.0
119389	9.0

series

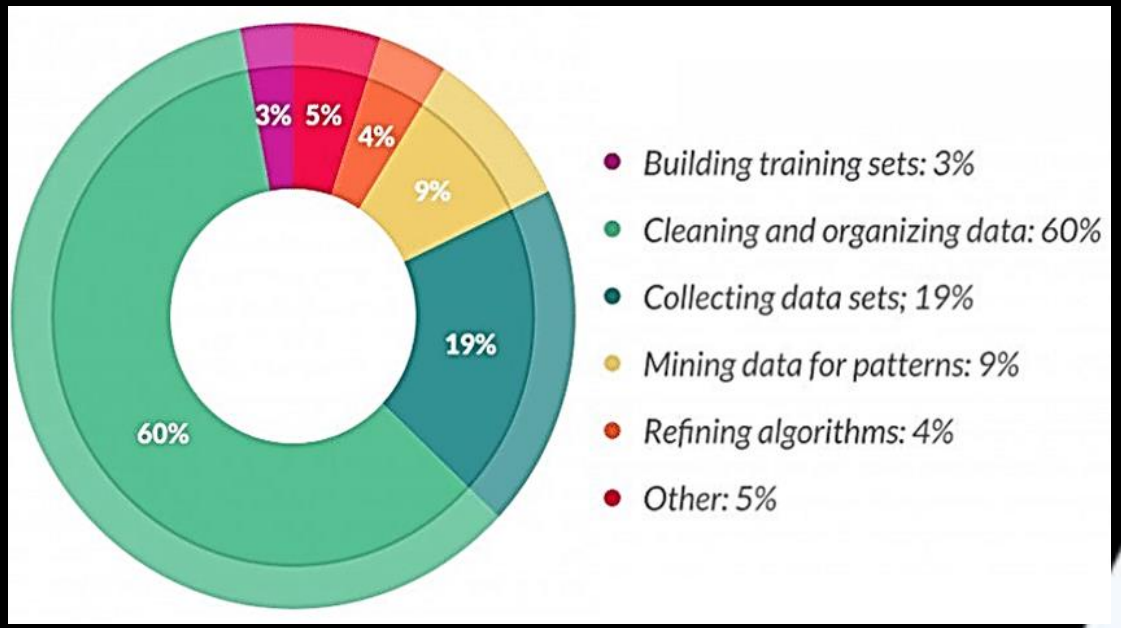
Basic Data Manipulation with

Why required required to preprocessing the data?



Preprocessing Datasets

The first step in your data science or machine learning workflow is cleaning data. Without clean data you'll be having a much harder time seeing the actual important parts in your exploration. According to Crowd flower, data scientist **spend 60%**.



Data manipulation with pandas

1. **Selecting**
2. **Filtering**
3. **Deleting**
4. **Sorting**

Data Selection

Selection by Columns

```
ritz_jager[['agent','agent','children']]
```

	agent	agent	children
0	9.0	9.0	0.0
1	9.0	9.0	0.0
2	9.0	9.0	0.0
3	304.0	304.0	0.0
4	240.0	240.0	0.0
...	...	...	...
119385	394.0	394.0	0.0
119386	9.0	9.0	0.0
119387	9.0	9.0	0.0
119388	89.0	89.0	0.0
119389	9.0	9.0	0.0

119390 rows × 3 columns

Selection by Rows

```
ritz_jager.iloc[::2]
```

	hotel_type	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_
0	Resort Hotel	0	342	2015	July	
2	Resort Hotel	0	7	2015	July	
4	Resort Hotel	0	14	2015	July	
6	Resort Hotel	0	0	2015	July	
8	Resort Hotel	1	85	2015	July	
...	...	...	...	...	...	
119380	City Hotel	0	44	2017	August	
119382	City Hotel	0	135	2017	August	
119384	City Hotel	0	21	2017	August	
119386	City Hotel	0	102	2017	August	
119388	City Hotel	0	109	2017	August	

59695 rows × 32 columns

Data Selection

Selection .loc

```
ritz_jager.loc[1,['agent','agent','country_origin','children']]
```

```
agent              9
agent              9
country_origin   PRT
children           0
Name: 1, dtype: object
```

Selection .iloc

```
ritz_jager.iloc[::2]
```

	hotel_type	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_
0	Resort Hotel	0	342	2015	July	
2	Resort Hotel	0	7	2015	July	
4	Resort Hotel	0	14	2015	July	
6	Resort Hotel	0	0	2015	July	
8	Resort Hotel	1	85	2015	July	
...	...	...	...	...	...	
119380	City Hotel	0	44	2017	August	
119382	City Hotel	0	135	2017	August	
119384	City Hotel	0	21	2017	August	
119386	City Hotel	0	102	2017	August	
119388	City Hotel	0	109	2017	August	

59695 rows × 32 columns

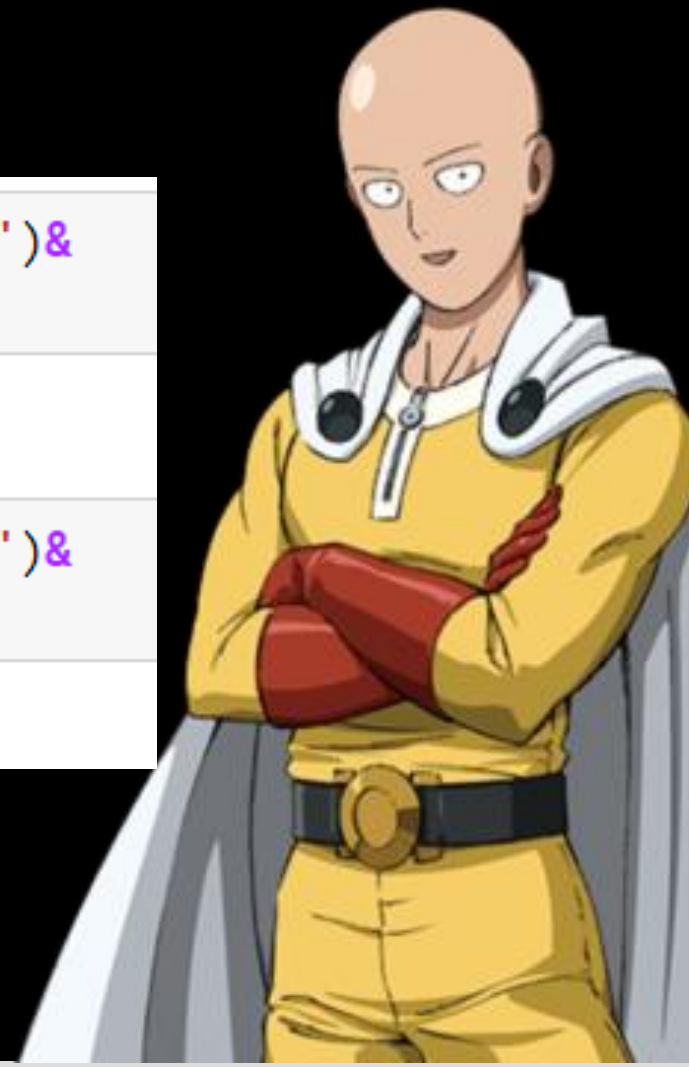
Data Filtering

```
len(ritz_jager[(ritz_jager['arrival_date_month']=='July')&
               (ritz_jager['stays_in_week_nights']>3)])
```

```
3838
```

```
len(ritz_jager[(ritz_jager['arrival_date_month']=='June')&
               (ritz_jager['stays_in_week_nights']>4)])
```

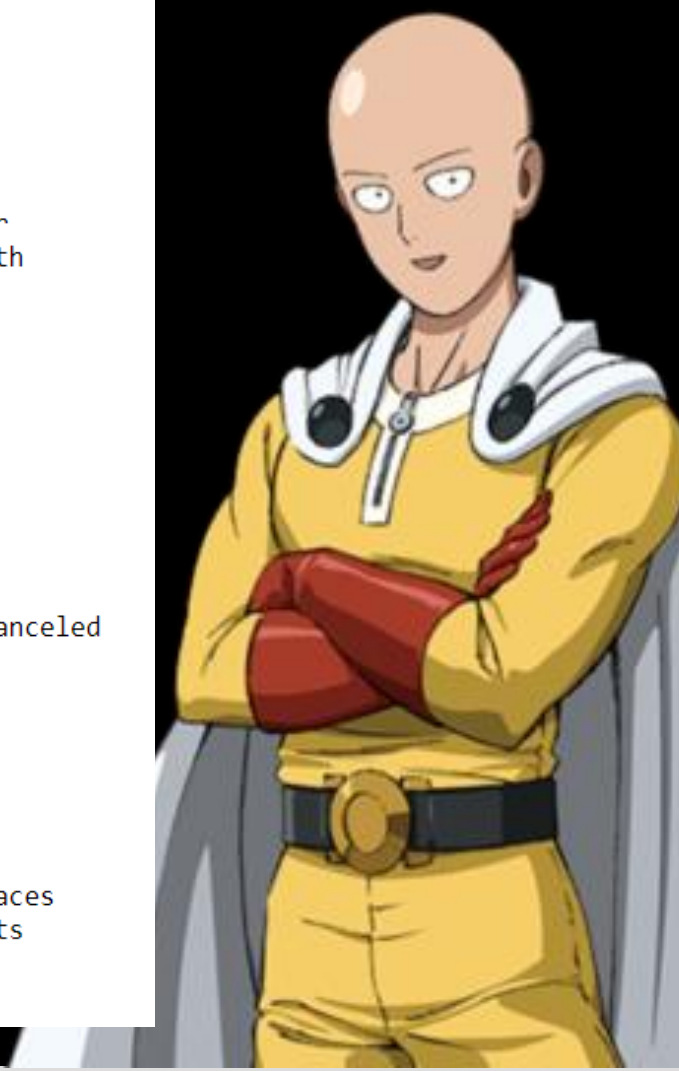
```
1678
```



Data Deletion

```
# 1. drop company column
ritz_jager = ritz_jager.drop('company',axis = 1)
```

```
agent
children
reservation_status_date
market_segment
is_canceled
lead_time
arrival_date_year
arrival_date_month
arrival_date_week_number
arrival_date_day_of_month
stays_in_weekend_nights
stays_in_week_nights
adults
babies
meal_type
country_origin
distribution_channel
reservation_status
is_repeated_guest
previous_cancellations
previous_bookings_not_canceled
reserved_room_type
assigned_room_type
booking_changes
deposit_type
days_in_waiting_list
customer_type
adr
required_car_parking_spaces
total_of_special_requests
hotel_type
dtype: int64
```



Data Sorting

```
s = ritz_jager[['arrival_date_month','is_canceled']][ok]
s.groupby('arrival_date_month').count().reset_index().sort_values('is_canceled',ascending = False)
```

	arrival_date_month	is_canceled
1	August	8638
5	July	7919
8	May	7114
10	October	6914
7	March	6645
0	April	6565
6	June	6404
11	September	6392
3	February	5372
9	November	4672
2	December	4409
4	January	4122

Advanced Data Manipulation With Pandas

1. Pivoting
2. Crosstab
3. Binning
4. Categorical Encoding

Pivoting

pivot table is a table of statistics that summarizes the data of a more extensive table. in practical terms, a pivot table calculates a statistics on abreakdown of values. for the first column, it displays values as rows and for the second columns as columns

```
data = ritz_jager[ritz_jager['market_segment'] == 'Online TA']
pd.pivot_table(data, index = ['customer_type','country_origin'],
               values='adr', aggfunc='mean').head(10)
```

		adr
customer_type	country_origin	
Contract	AGO	124.818571
	ARG	77.210000
	AUS	113.633333
	AUT	106.863000
	AZE	112.500000
	BEL	112.400357
	BGR	112.500000
	BLR	107.166667
	BRA	99.140000
	CHE	104.769500

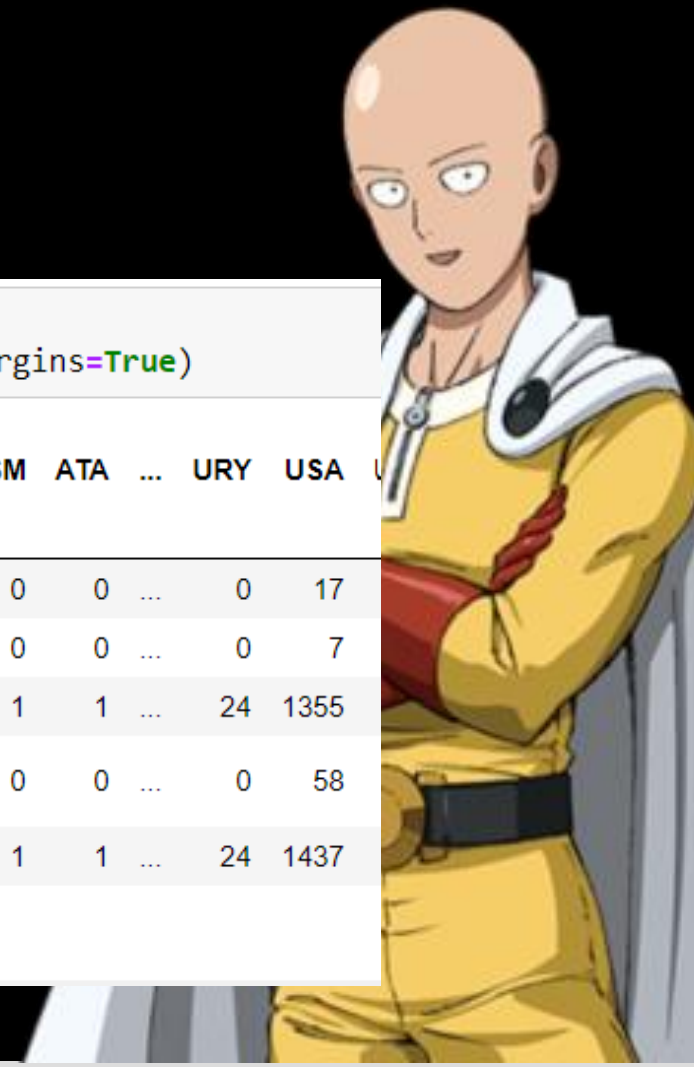
Crosstab

Compute a simple cross-tabulation of two or more factors. By default computes a frequency table of the factors

```
#crosstab
pd.crosstab(data.customer_type, data.country_origin, margins=True)
```

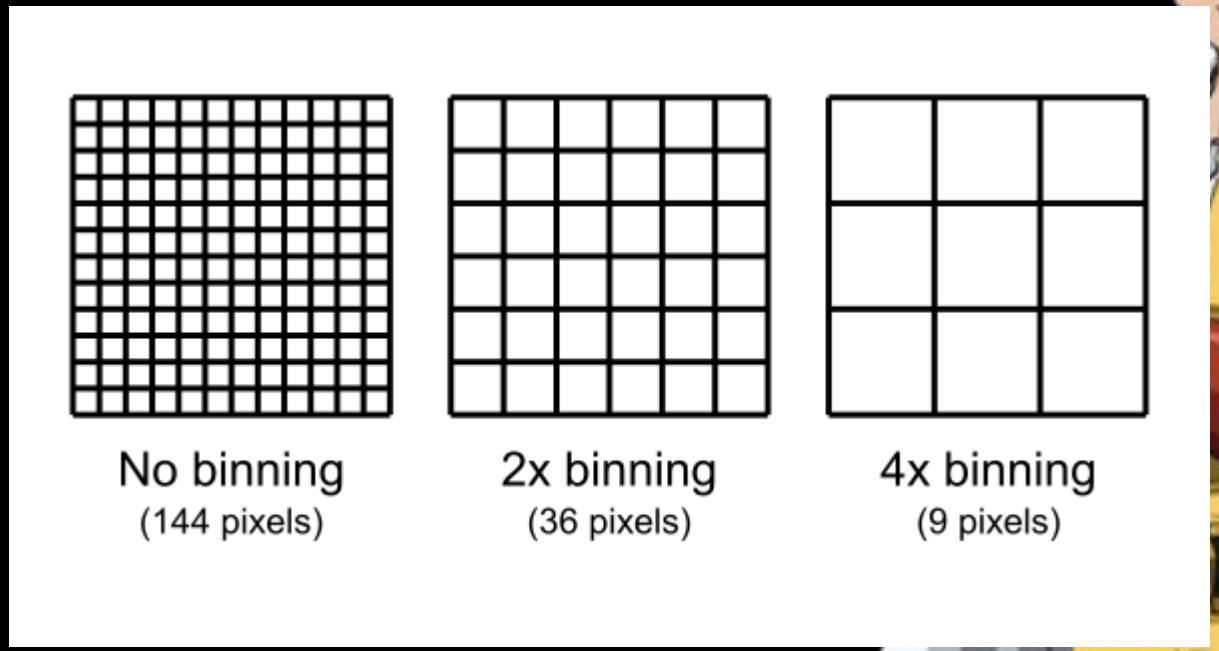
country_origin customer_type	ABW	AGO	AIA	ALB	AND	ARE	ARG	ARM	ASM	ATA	...	URY	USA
Contract	0	7	0	0	0	0	1	0	0	0	...	0	17
Group	0	0	0	0	0	0	1	0	0	0	...	0	7
Transient	2	196	1	3	6	46	142	4	1	1	...	24	1355
Transient-Party	0	1	0	2	0	1	10	0	0	0	...	0	58
All	2	204	1	5	6	47	154	4	1	1	...	24	1437

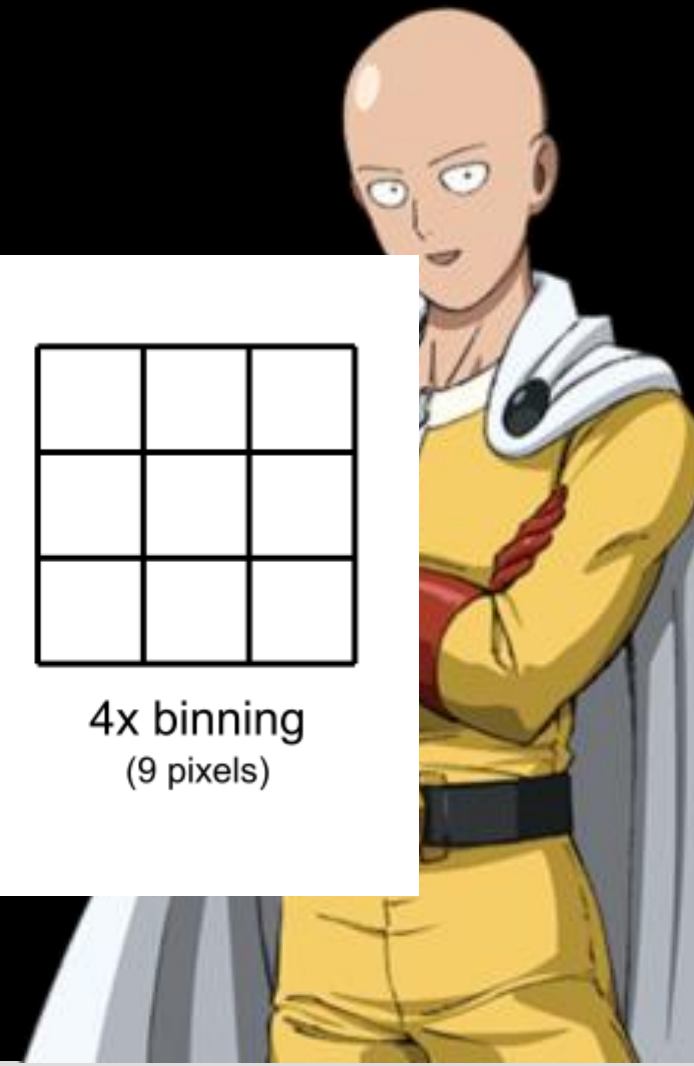
5 rows × 167 columns



Binning

1. When dealing with continuos numeric data, it is often helpful to bin the data into multiple buckets for further analysis





Categorical Encoding

We doing encoding so that machine learning can understanding the data.

If we have ordinal data we have to label encoding with numeric data.

If we have nominal data we have to one hot encoding.

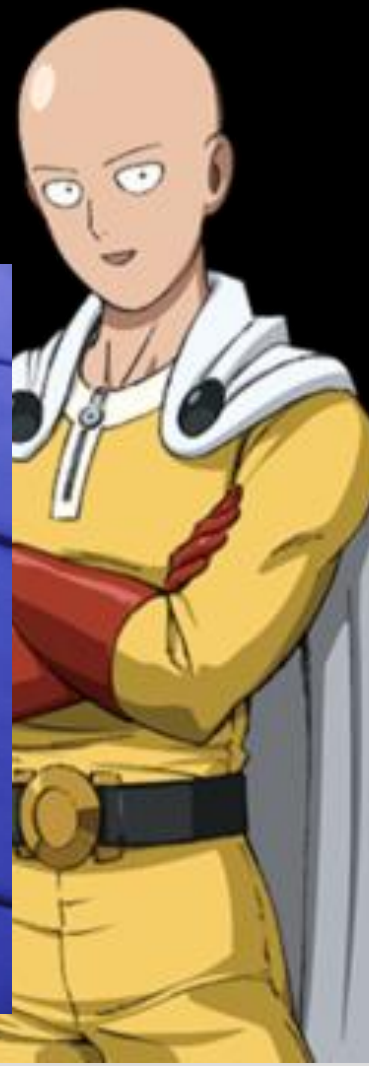
Data Cleansing

1. Missing Values Checking and handling
2. Duplicates Checking
3. Anomaly and Outlier Detection
4. Data Type Checking
5. Data Type Correction
6. Feature Extraction

Missing Values Checking and Handling

Why missing value exist?

1. Missed data
2. Deleted
3. Corrupt
4. Mismatch row and column
5. real value not availabel



Cleaning & Transform Data

1. Importing library and datasets

Import Data

```
# import lybrary
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv('data.csv')
```

Cleaning & Transform Data

2. Shape, head, tail data

```
print(data.shape)
data.head()
```

```
(205, 26)
```

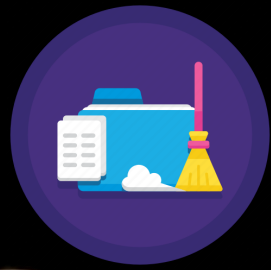
	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepo
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	10.0	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	8.0	

5 rows × 26 columns

```
data.tail()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepow
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	11
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	16
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	13
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.4	23.0	10
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	11

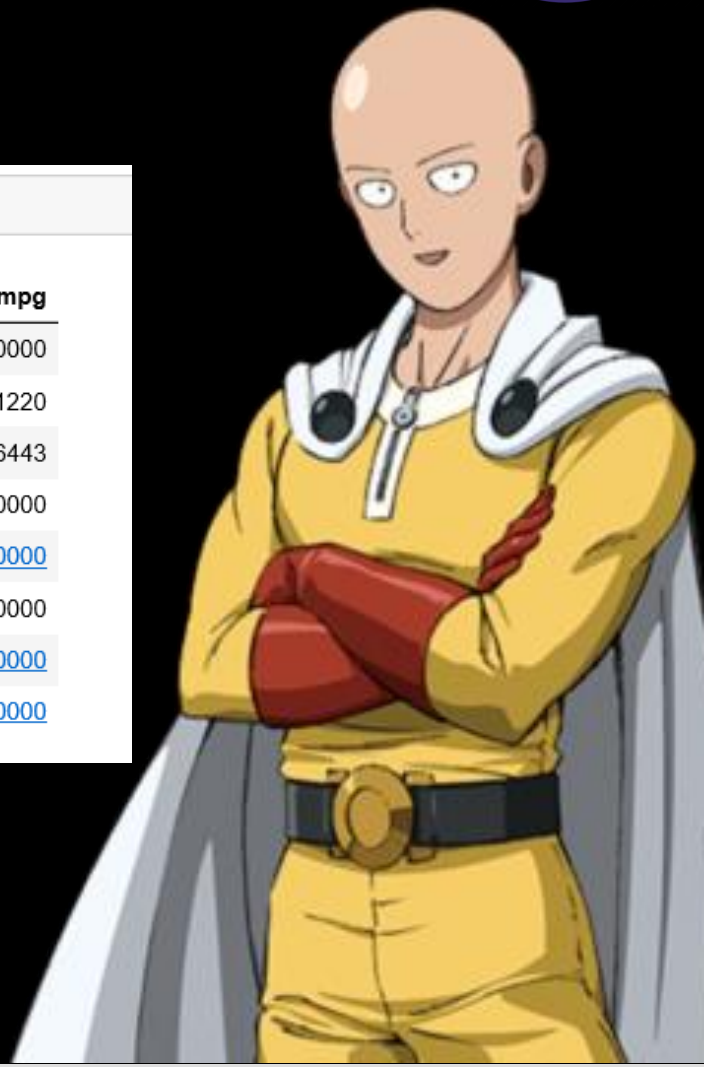
Cleaning & Transform Data



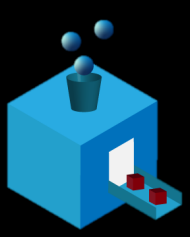
3. Describe data

```
data.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000



Cleaning & Transform Data



4. See data types

```
data.dtypes
```

```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

Cleaning & Transform Data

5. Fixing '?' values and see missing values

```
#identify missing value
data.replace('?', np.NaN, inplace = True)
data.isnull().sum().sort_values(ascending = False)
```

```
normalized-losses    41
price                 4
stroke                4
bore                  4
peak-rpm              2
horsepower            2
num-of-doors          2
length                0
make                  0
fuel-type             0
aspiration            0
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
height                0
width                 0
highway-mpg           0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
compression-ratio     0
city-mpg              0
symboling             0
dtype: int64
```

Find anomaly data

Cleaning & Transform Data

6. See percentage missing values

In column ***num-of-doors***

We can fixing with mode.
Because that's column is
categorical data.
But the other
missing values are numerical.
We have to check the
Distribution.

```
#presentage missing values
data.isnull().sum().sort_values(ascending = False)/len(data)
```

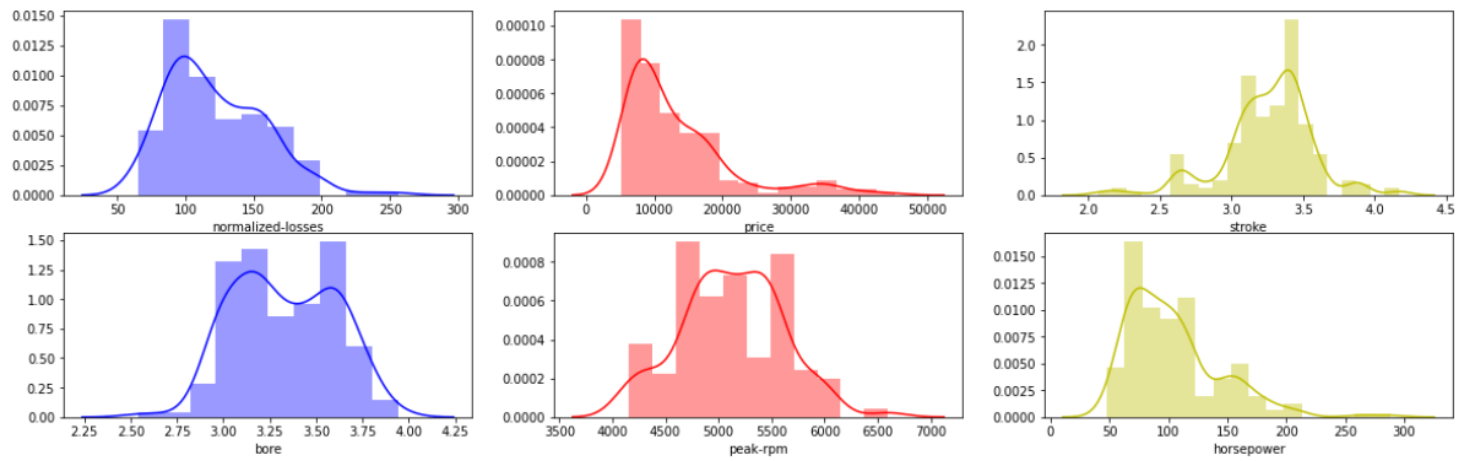
```
normalized-losses    0.200000
price                0.019512
stroke               0.019512
bore                 0.019512
peak-rpm             0.009756
horsepower           0.009756
num-of-doors         0.009756
length               0.000000
make                 0.000000
fuel-type            0.000000
aspiration           0.000000
body-style           0.000000
drive-wheels         0.000000
engine-location      0.000000
wheel-base           0.000000
height               0.000000
width                0.000000
highway-mpg          0.000000
curb-weight          0.000000
engine-type          0.000000
num-of-cylinders     0.000000
engine-size          0.000000
fuel-system          0.000000
compression-ratio    0.000000
city-mpg             0.000000
symboling            0.000000
dtype: float64
```

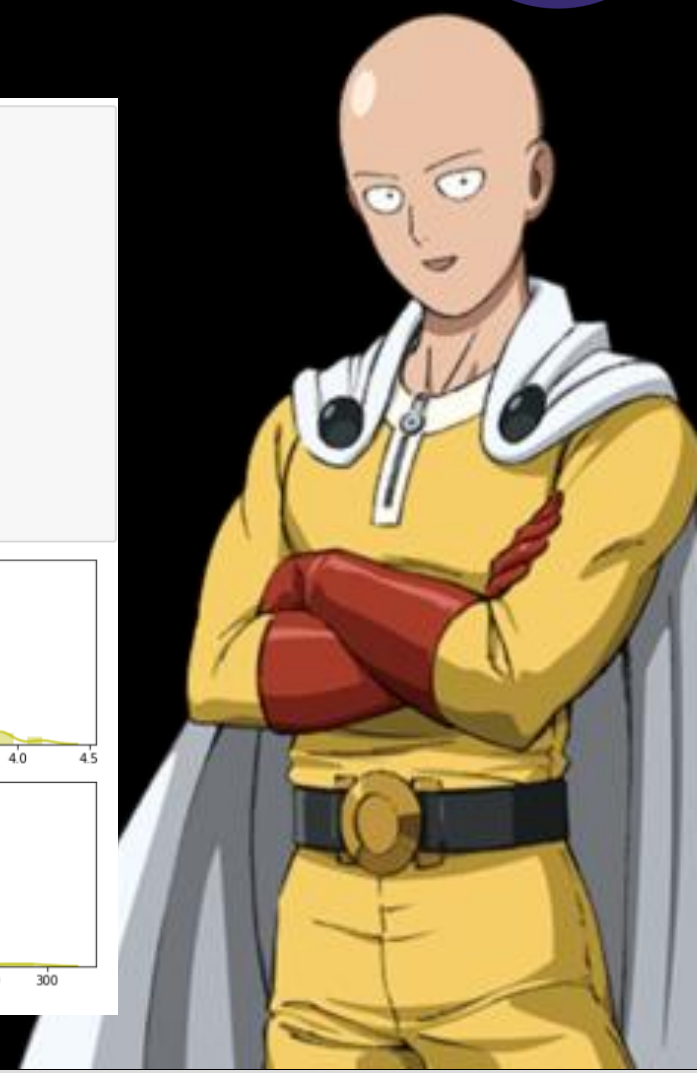
Cleaning & Transform Data

7. Handle missing value

In all column we can fillna with median. Because all graph show a tendency skew left and right. There is no perfect normal, So the representative value is ***median*** from each column

```
nl = data['normalized-losses']
price = data['price']
stroke = data['stroke']
bore = data['bore']
peak_rpm = data['peak-rpm']
horse= data['horsepower']

f, axes = plt.subplots(2,3, figsize = (20, 6))#,sharex = True,sharey=True)
sns.distplot(nl,color = 'b', ax = axes[0,0])
sns.distplot(price,color = 'r', ax = axes[0,1])
sns.distplot(stroke,color = 'y', ax = axes[0,2])
sns.distplot(bore,color = 'b',ax = axes[1,0])
sns.distplot(peak_rpm,color = 'r', ax = axes[1,1])
sns.distplot(horse,color = 'y', ax = axes[1,2])

plt.show()
```





Cleaning & Transform Data

8. Handling Missing Value

```
data['num-of-doors'].value_counts()
```

```
four    114
two      89
Name: num-of-doors, dtype: int64
```

In columns num-of-doors
Fillna with value:

"four"

Because it's a mode.

```
# replace with median in NaN
data= data.fillna(data.median())
data['num-of-doors'] = data['num-of-doors'].fillna('four')
data.isnull().sum().sort_values(ascending = False)
```

```
price                0
highway-mpg          0
normalized-losses    0
make                 0
fuel-type            0
aspiration           0
num-of-doors         0
body-style           0
drive-wheels         0
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
bore                 0
stroke               0
compression-ratio    0
horsepower           0
peak-rpm             0
city-mpg             0
symboling            0
dtype: int64
```

All clean

Convert data types to datetime

```
# convert datetime
from datetime import datetime
ritz_jager['date_reservation_status'] = pd.to_datetime(ritz_jager['reservation_status_date'],format = '%d/%m/%Y')
```

```
adr                             float64
required_car_parking_spaces       int64
total_of_special_requests         int64
reservation_status               object
reservation_status_date          object
dtype: object
```

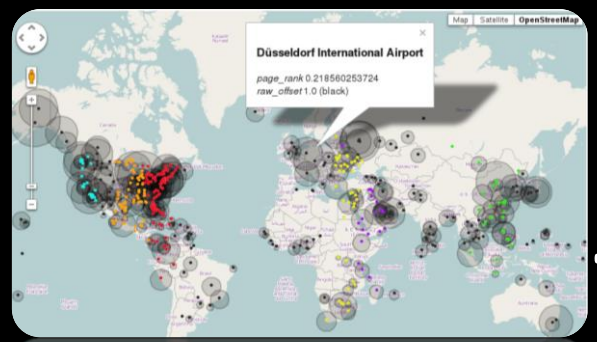
```
adr                                    float64
required_car_parking_spaces              int64
total_of_special_requests                int64
reservation_status                      object
reservation_status_date                 object
date_reservation_status          datetime64[ns]
dtype: object
```

How to make data visualization with Python?

We Visualize data with Matplotlib and Seaborn or folium to map visualization

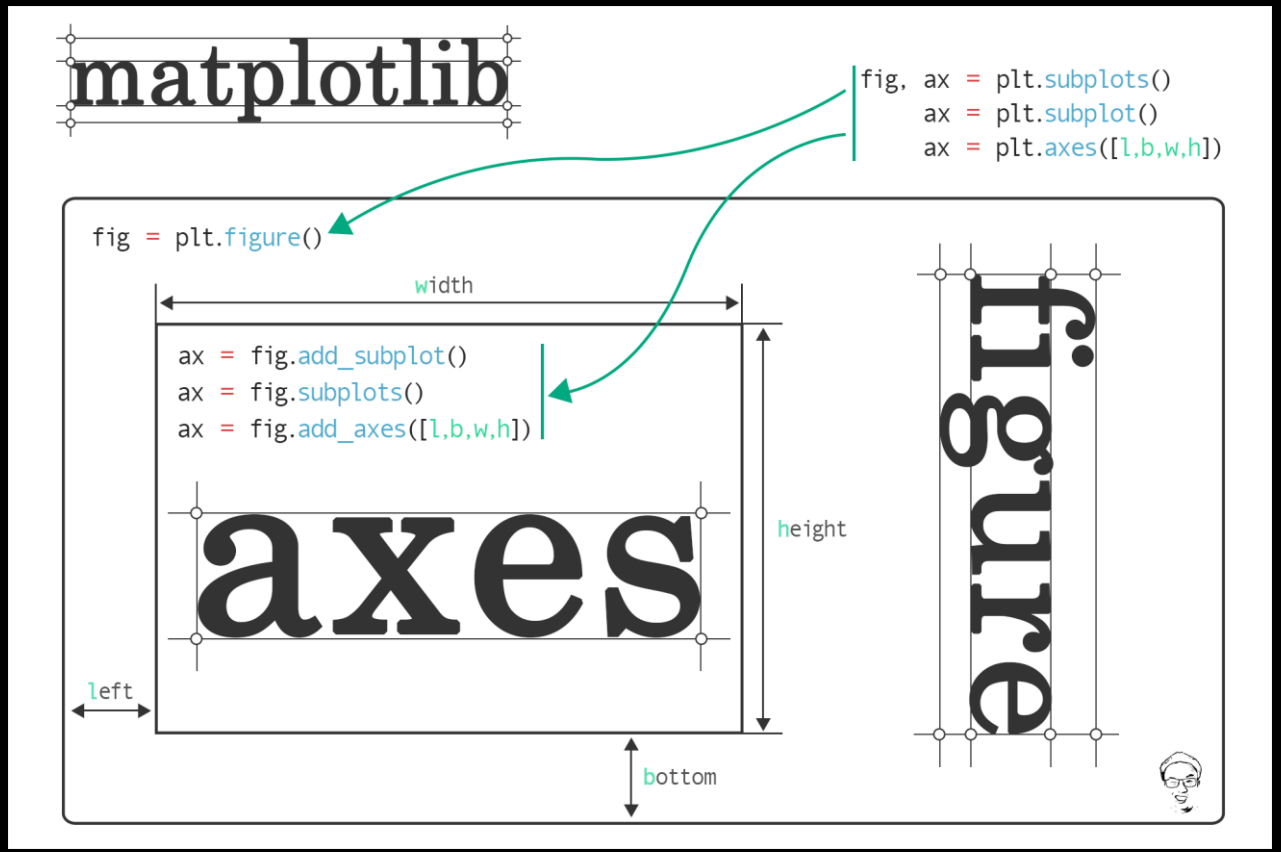
matplotlib

Seaborn

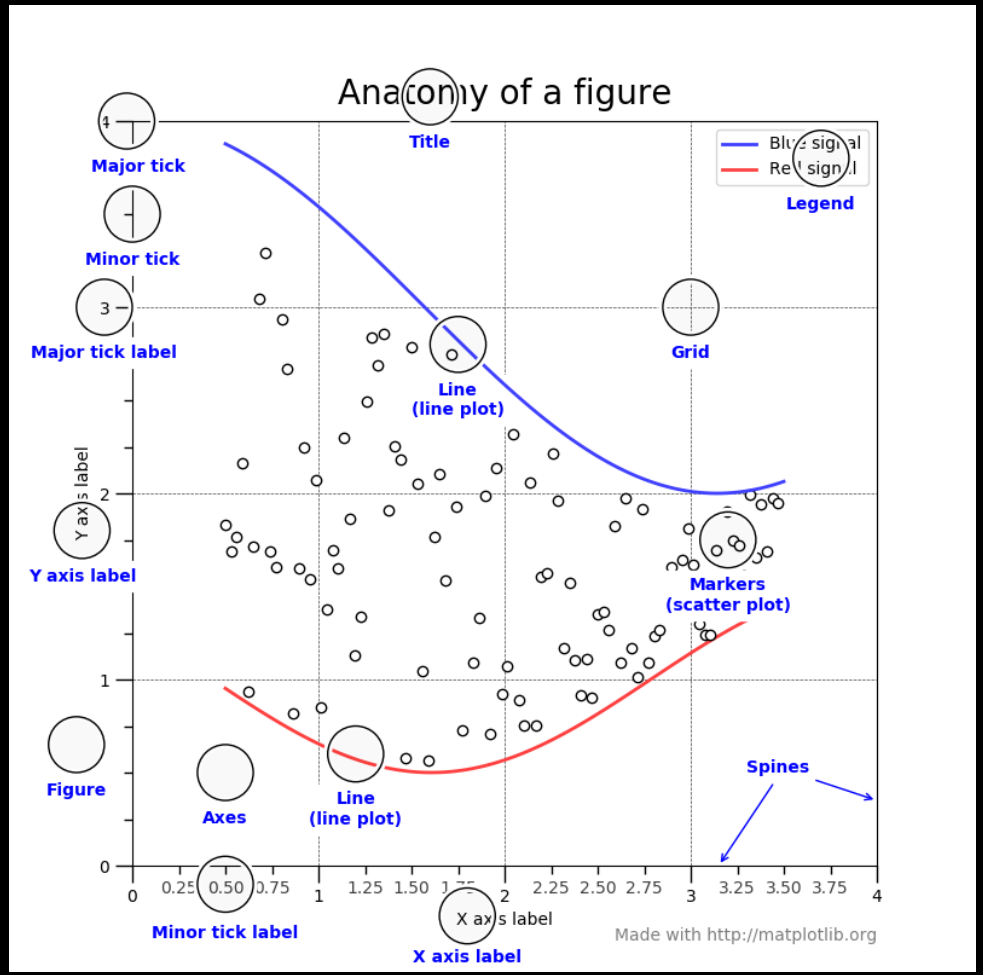


folium

Parts of Visualization in Python



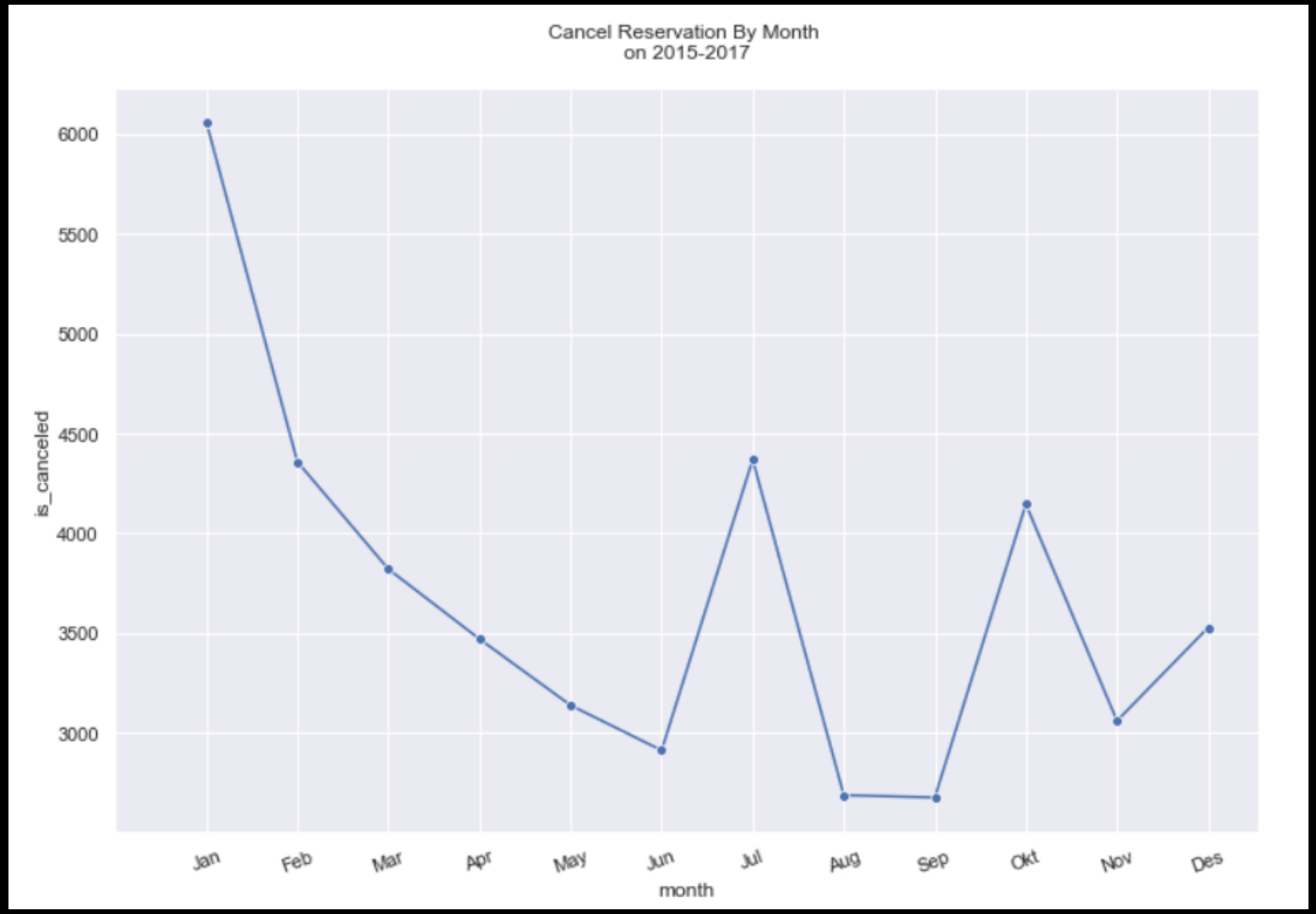
Anatomy of a figure



Kinds of Data Visualization

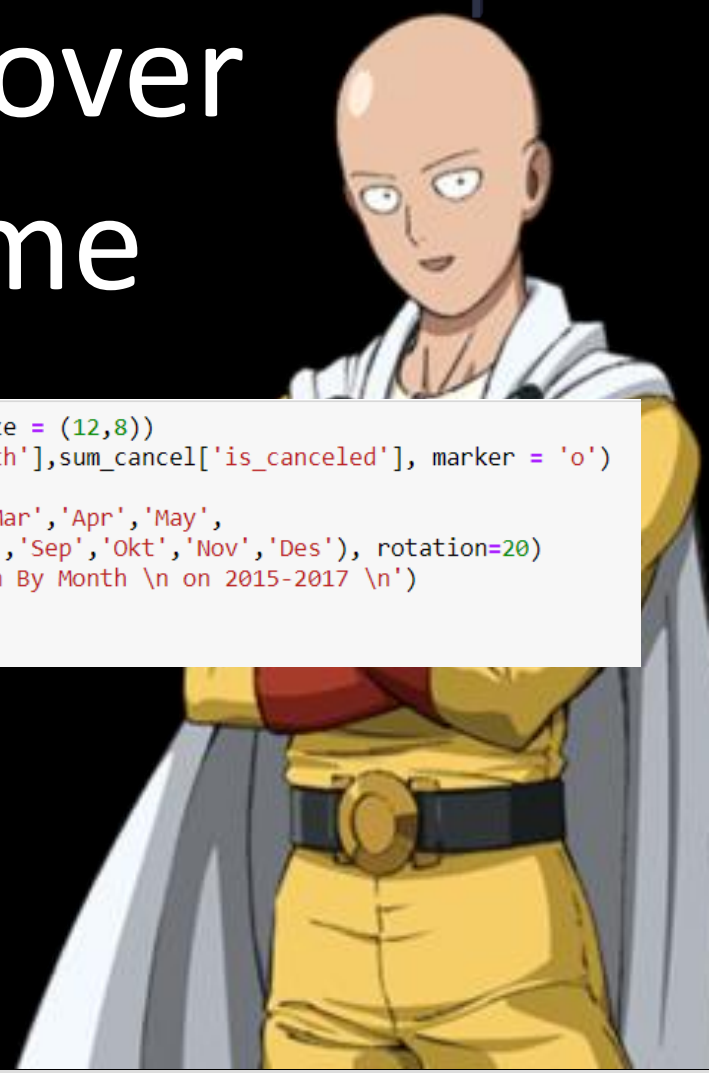
1. Line Chart
2. Bar chart
3. Histogram chart
4. Boxplot
5. Pie Chart
6. Heatmap
7. Scatter Plot

Line Graph



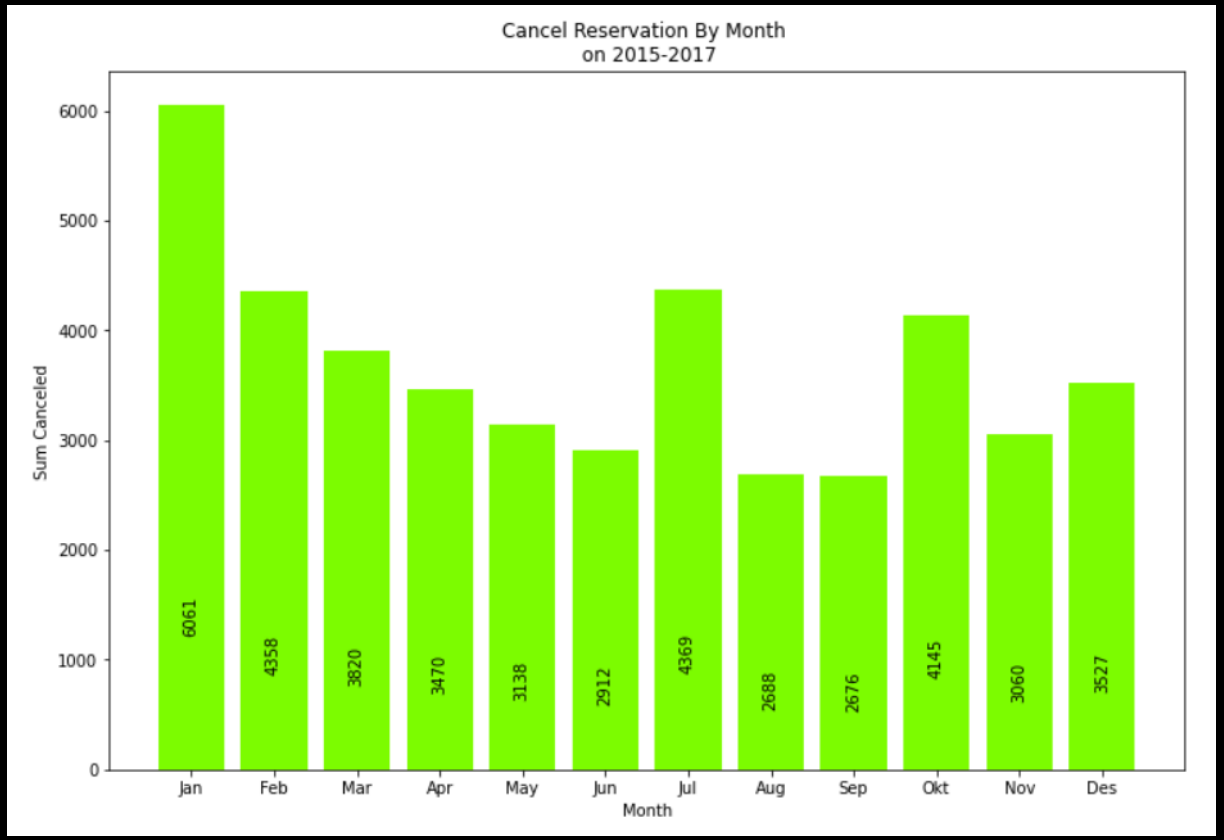
See over time

```
ax, fig = plt.subplots(figsize = (12,8))
sns.lineplot(sum_cancel['month'],sum_cancel['is_canceled'], marker = 'o')
plt.xticks(np.arange(13),
           ('','Jan','Feb','Mar','Apr','May',
            'Jun','Jul','Aug','Sep','Okt','Nov','Des'), rotation=20)
plt.title('Cancel Reservation By Month \n on 2015-2017 \n')
sns.set()
plt.show()
```



Bar Plot

To Compare

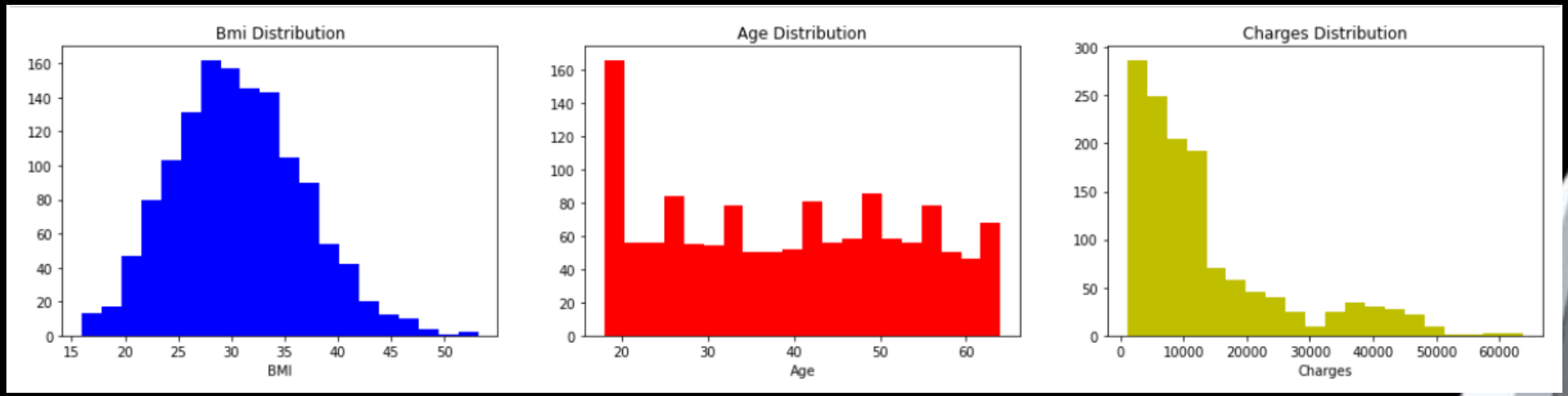



```
ax, fig = plt.subplots(figsize = (12,8))
bar_plots = plt.bar(sum_cancel['month'],sum_cancel['is_canceled'],
                    color = 'lawngreen', tick_label = ['Jan','Feb','Mar','Apr',
                                                       'May','Jun','Jul','Aug',
                                                       'Sep','Okt','Nov','Des'])
#plt.xticks(sum_cancel['month'])
plt.title('Cancel Reservation By Month \n on 2015-2017')
plt.xlabel('Month')
def autolabel(rects):
    for idx,rect in enumerate(bar_plots):
        height = rect.get_height()
        plt.text(rect.get_x() + rect.get_width()/2., 0.2*height,
                 sum_cancel['is_canceled'][idx],
                 ha='center', va='bottom', rotation=90)
autolabel(bar_plots)
plt.ylabel('Sum Canceled')
plt.show()
```

Histogram Plot

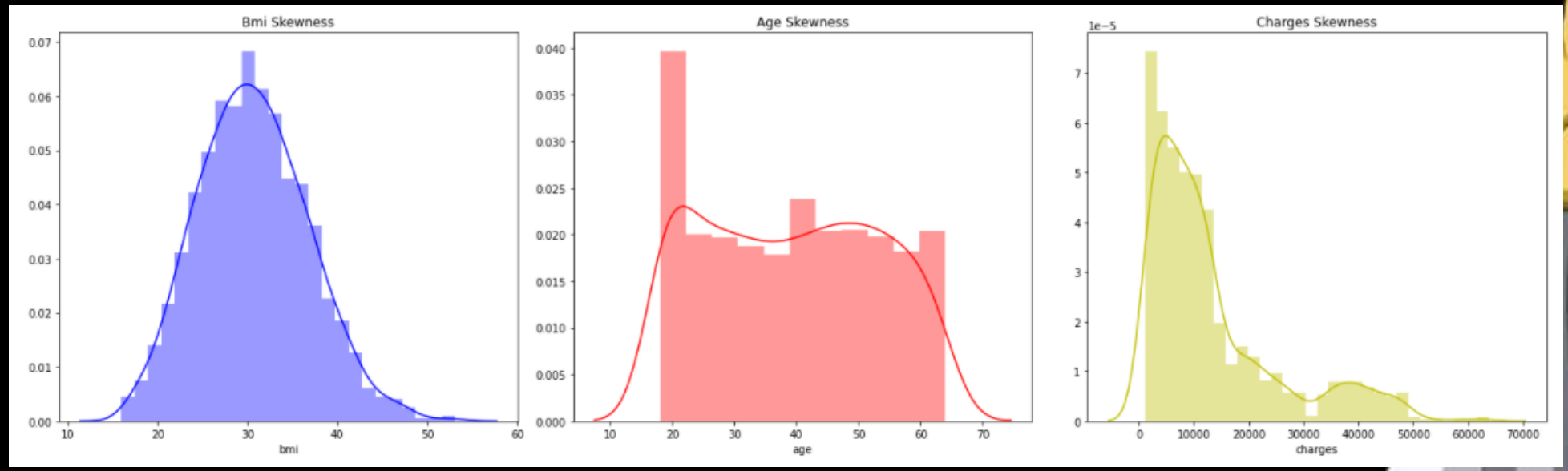
```
bmi = insurance['bmi']
age = insurance['age']
charges = insurance['charges']
fig, ax = plt.subplots(ncols = 3, figsize = (20, 4))
ax[0].hist(bmi,color = 'b',bins = 20)
ax[1].hist(age,color = 'r', bins = 20)
ax[2].hist(charges,color = 'y',bins = 20)
ax[0].set_title('Bmi Distribution')
ax[1].set_title('Age Distribution')
ax[2].set_title('Charges Distribution')
ax[0].set_xlabel('BMI')
ax[1].set_xlabel('Age')
ax[2].set_xlabel('Charges')

plt.show()
```

See distribution



Seeing Skew Plot

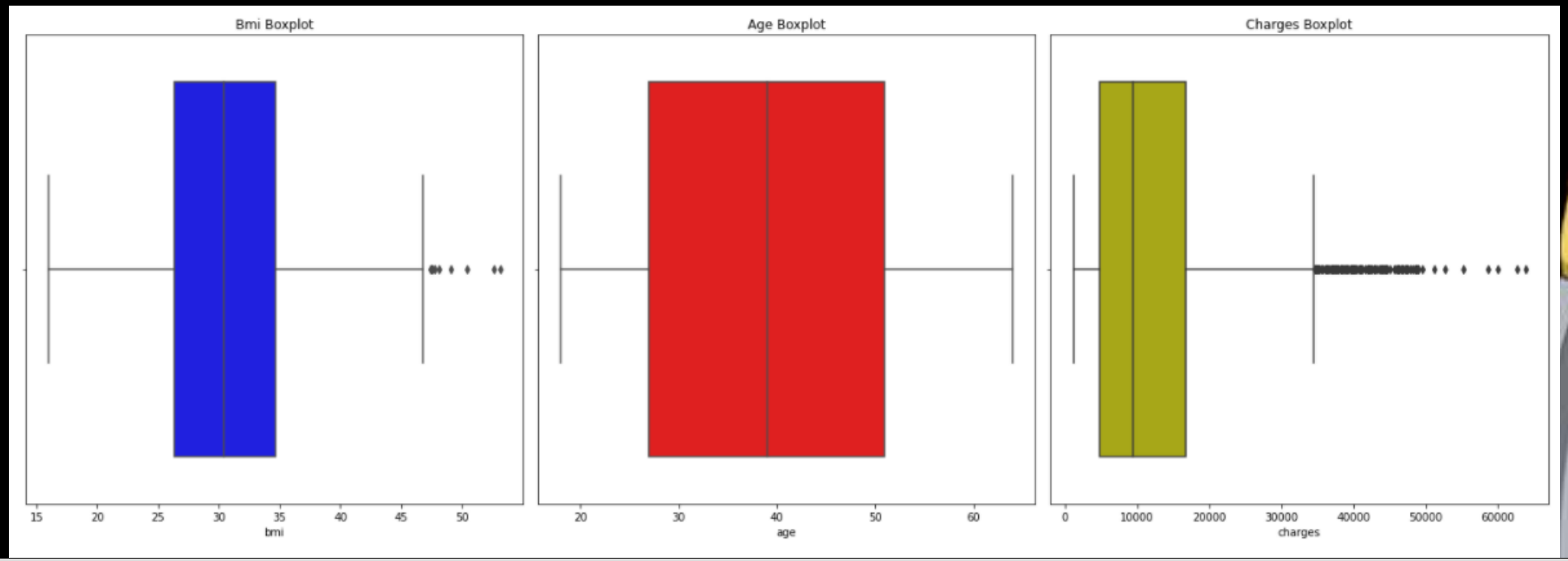
```
f, axes = plt.subplots(1,3, figsize = (20, 6))
#sns.despine(left=True)
sns.distplot(insurance['bmi'], color = 'b',ax = axes[0])
sns.distplot(insurance['age'], color = 'r', ax = axes[1])
sns.distplot(insurance['charges'], color = 'y',ax = axes[2])
#plt.setp(axes, yticks=[])
axes[0].set_title('Bmi Skewness')
axes[1].set_title('Age Skewness')
axes[2].set_title('Charges Skewness')
plt.tight_layout()
print(f"score skewness bmi: {skew(insurance['bmi'])}")
print(f"score skewness age: {skew(insurance['age'])}")
print(f"score skewness charges: {skew(insurance['charges'])}")
```



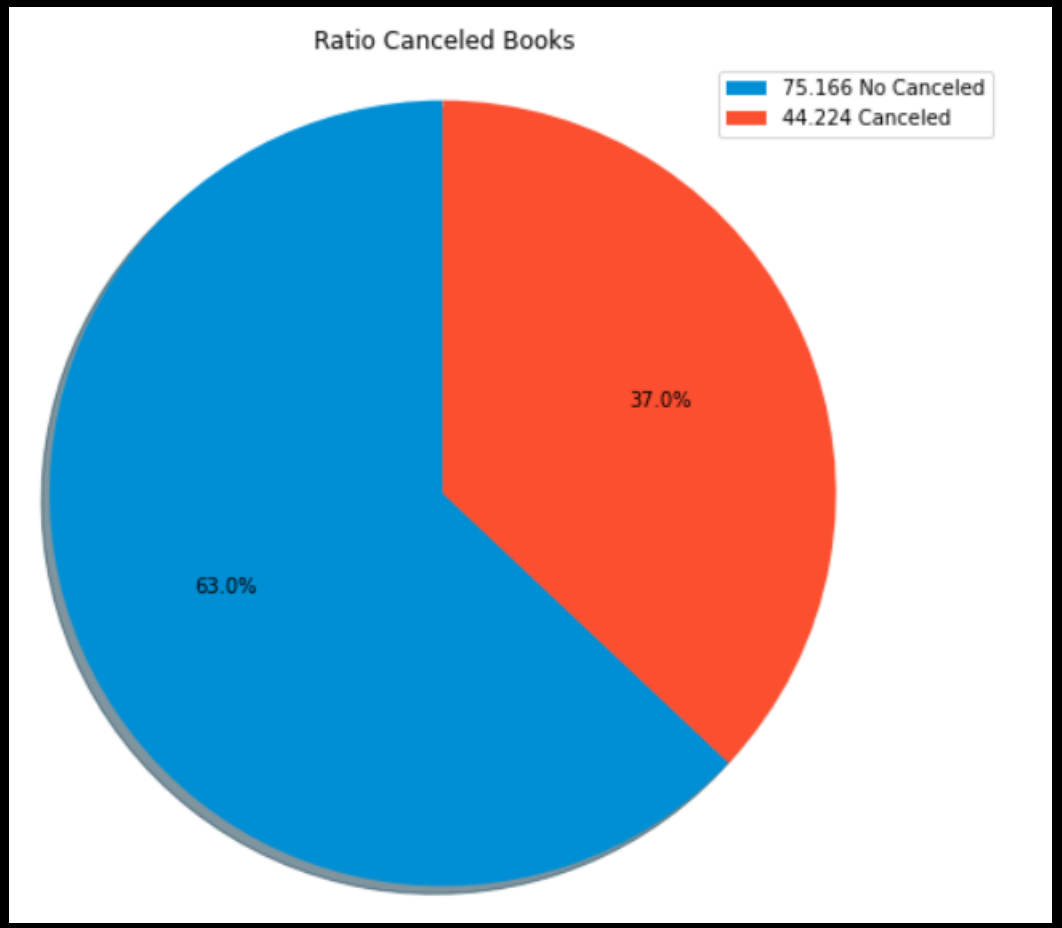
Box Plot Chart

```
f, axes = plt.subplots(1,3, figsize = (20, 7))
#sns.despine(left=True)
sns.boxplot(insurance['bmi'], color = 'b',ax = axes[0])
sns.boxplot(insurance['age'], color = 'r', ax = axes[1])
sns.boxplot(insurance['charges'], color = 'y',ax = axes[2])
axes[0].set_title('Bmi Boxplot')
axes[1].set_title('Age Boxplot')
axes[2].set_title('Charges Boxplot')
#plt.setp(axes, yticks=[])
plt.tight_layout()
```

See outlier



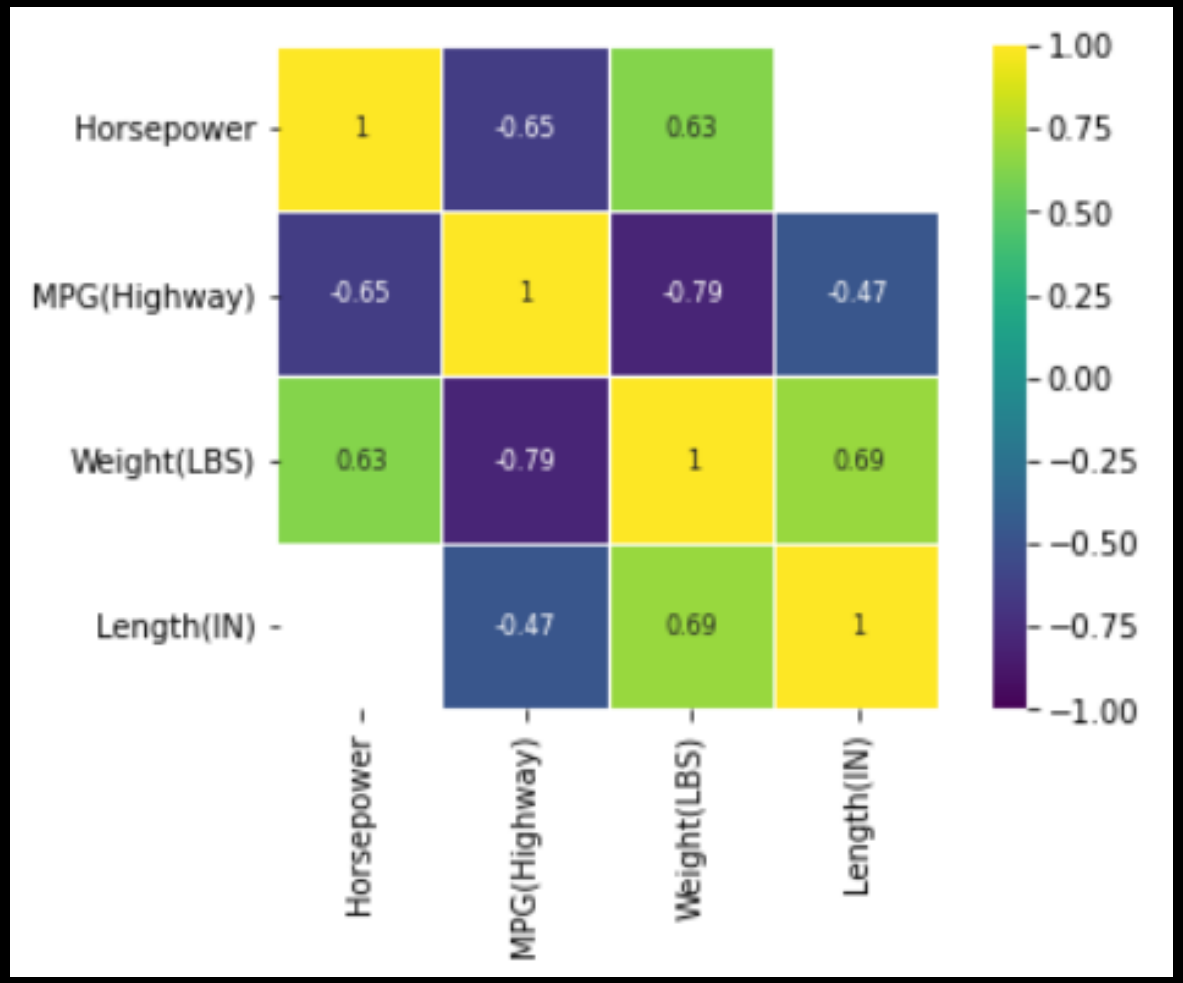
Pie Chart



See Ratio

```
fig, ax = plt.subplots(figsize = (10,8))
colors = ['#008fd5', '#fc4f30']
sizes = s[:] # your data
recipe = ["75.166 No Canceled",
          "44.224 Canceled"]
plt.pie(sizes, colors=colors, shadow=True,
                            startangle=90,autopct='%1.1f%%')

plt.title('Ratio Canceled Books')
plt.legend(labels = recipe, loc="upper right")
plt.axis('equal')
plt.show()
```

Heatmap Chart

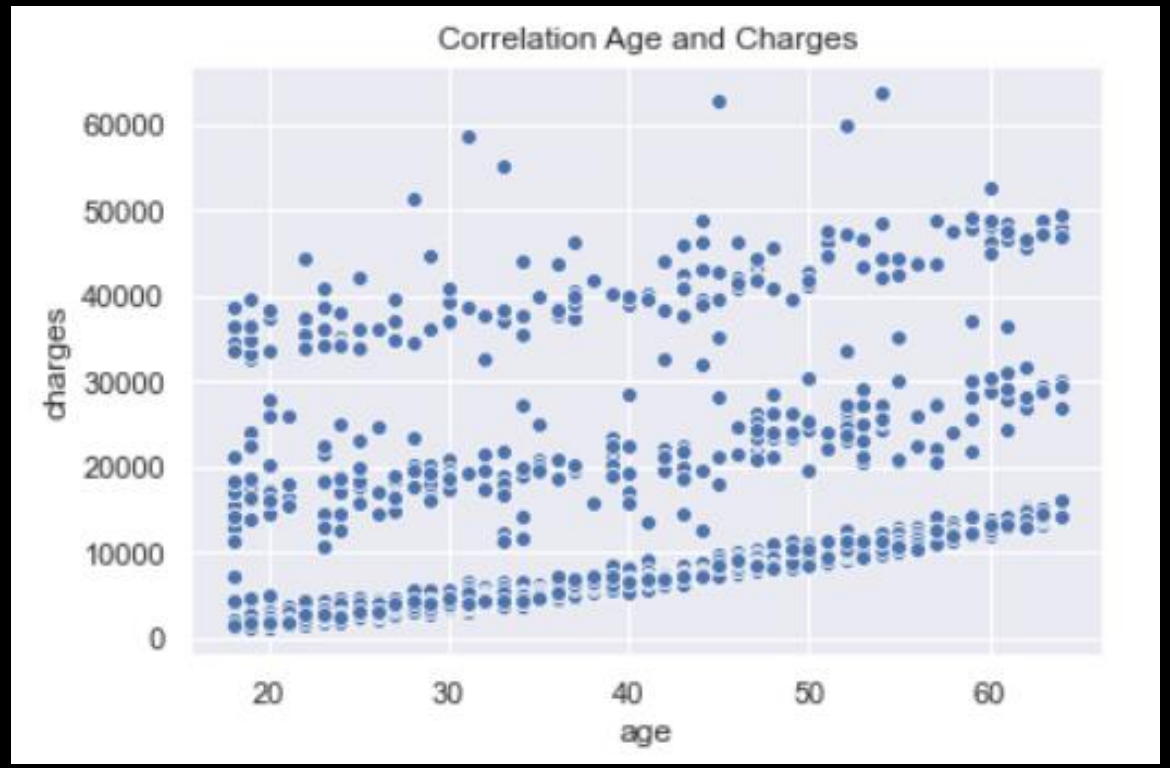


To Composition

```
fig, axes = plt.subplots(figsize = (6,4))
corr = cars_num.drop('MPG(City)', axis = 1).corr()
sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.4)], cmap = 'viridis',
            vmax = 1.0, vmin = -1.0, linewidths=0.1,
            annot=True, annot_kws={"size":8}, square=True);
```

Scatter Plot

```
f, axes = plt.subplots()
sns.scatterplot(insurance['age'],insurance['charges'])
axes.set_title('Correlation Age and Charges')
sns.set()
```



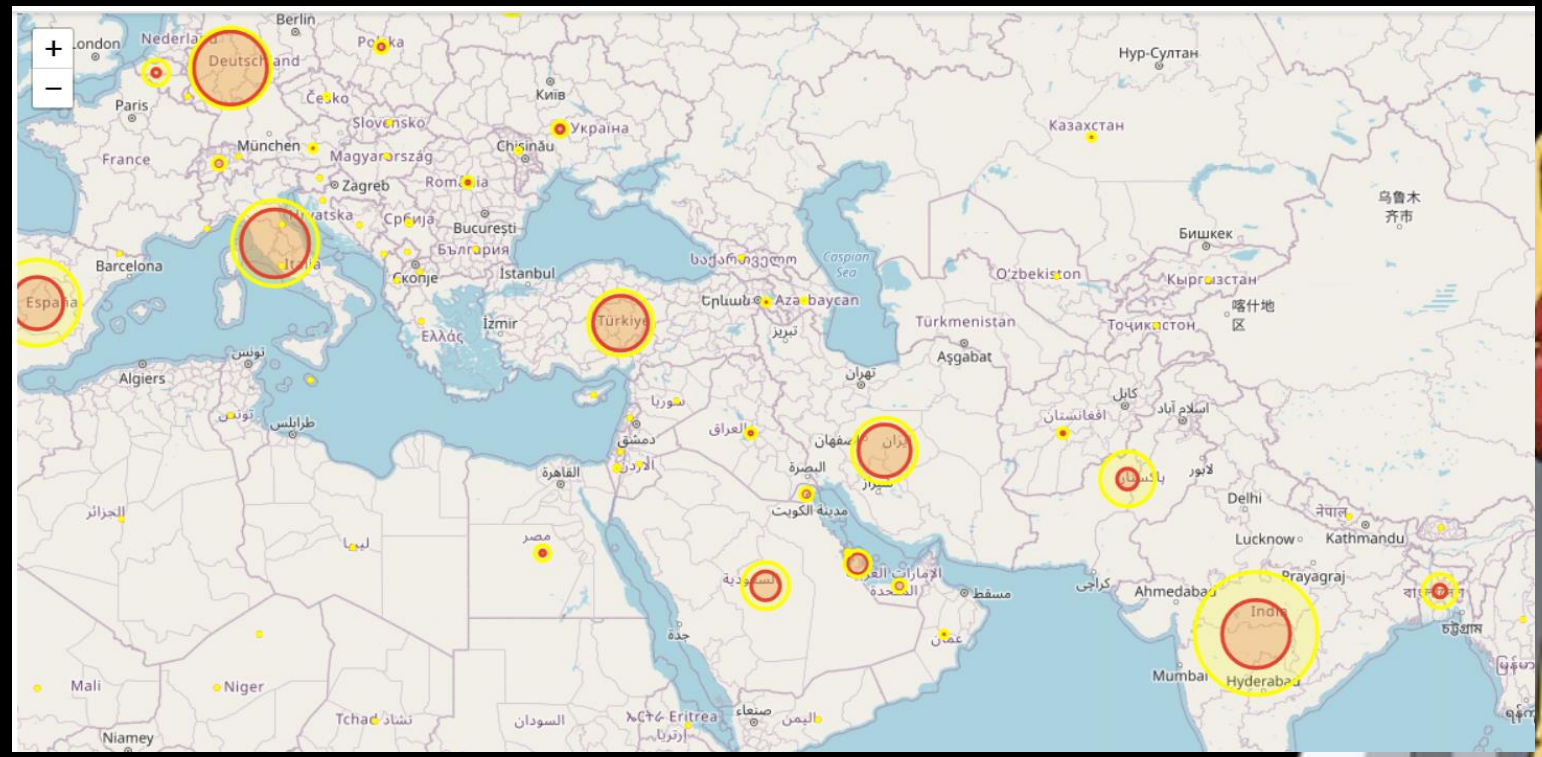
To See Correlation

Basic Map Chart

```
import folium
def generateBaseMap(default_location=[-7.8682695, 120.33345], default_zoom_start=4):
    base_map = folium.Map(location=default_location,
                          control_scale=True,
                          zoom_start=default_zoom_start)
    return base_map
base_map=generateBaseMap()
base_map
```

Covid Map Chart

```
# positive maping
base_map=generateBaseMap()
for i in range(0, len(df_global)):
    folium.Circle(location=[df_global.iloc[i]['latitude'], df_global.iloc[i]['longitude']],
                  radius=float(df_global.iloc[i]['sembuh']),#the bigger the positive, the bigger the circle
                  #popup = df_global.iloc[i], #so that the information appears when clicked
                  color='crimson',#with color
                  fill=True, #the circle inside also has a color
                  fill_color = 'crimson').add_to(base_map)
for i in range(0, len(df_global)):
    folium.Circle(location=[df_global.iloc[i]['latitude'], df_global.iloc[i]['longitude']],
                  radius=float(df_global.iloc[i]['positif']),
                  #popup = df_global.iloc[i],
                  color='yellow',
                  fill=True,
                  fill_color = 'yellow').add_to(base_map)
# Circle =  making bullets a number of countries because of using len (df_global)
base_map
```



Exploratory Data Analysis

INFOGRAPHICS

Data

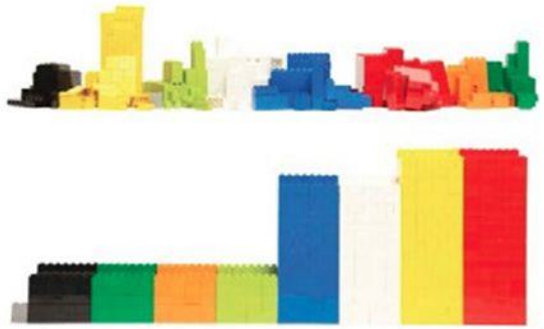
Your main contents

Sorted

Understand relationships and every small details

Arranged

Have a distribution plan, be creative & narrative

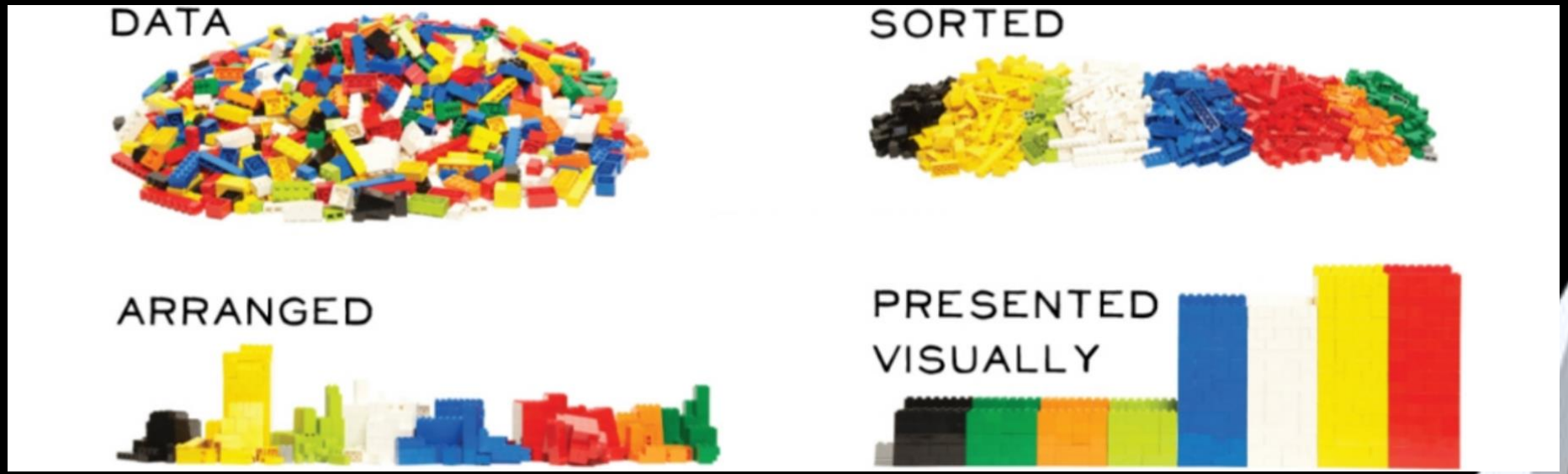


Presented Visually

Make the impact, there are free tolls available

Exploratory Data Analysis

EDA is process of studying
and investigating to find anomalies,
patterns, with statistical processes.



Why we have to apply EDA?

Data Scientist

EDA

Ensure the integrity of your gathered data and performed analysis

Remember concept GIGO

3 Parts of EDA

1. Cleansing
2. Defining questions
3. Visualizations

Source

1. https://www.guru99.com/what-is-data-analysis.html
2. http://www.data-analysis-in-python.org/#:~:text=Python%20is%20an%20increasingly%20popular,have%20accumulated%20over%20the%20years.
3. https://seaborn.pydata.org/
4. https://matplotlib.org/
5. https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html

Thank You