```
import seaborn as sns
df = sns.load_dataset("flights")
df
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.reset_index(drop="first",inplace=True)
df
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df["year"].unique() #returns unique elements of a particular column in dataframe
```

```
array([1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959,
       1960])
```

```
for i in df.columns :
  print(df[i].unique());
```

```
[1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960]
['Jan', 'Feb', 'Mar', 'Apr', 'May', ..., 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
Length: 12
Categories (12, object): ['Jan', 'Feb', 'Mar', 'Apr', ..., 'Sep', 'Oct', 'Nov', 'Dec']
[112 118 132 129 121 135 148 136 119 104 115 126 141 125 149 170 158 133
 114 140 145 150 178 163 172 199 184 162 146 166 171 180 193 181 183 218
 230 242 209 191 194 196 236 235 229 243 264 272 237 211 201 204 188 227
 234 302 293 259 203 233 267 269 270 315 364 347 312 274 278 284 277 317
 313 318 374 413 405 355 306 271 301 356 348 422 465 467 404 305 336 340
 362 363 435 491 505 359 310 337 360 342 406 396 420 472 548 559 463 407
 417 391 419 461 535 622 606 508 390 432]
```

```
for i in df.columns:
  if i == "year" or i == "passengers":
    print(df[i].unique())
```

```
[1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960]
[112 118 132 129 121 135 148 136 119 104 115 126 141 125 149 170 158 133
 114 140 145 150 178 163 172 199 184 162 146 166 171 180 193 181 183 218
 230 242 209 191 194 196 236 235 229 243 264 272 237 211 201 204 188 227
 234 302 293 259 203 233 267 269 270 315 364 347 312 274 278 284 277 317
 313 318 374 413 405 355 306 271 301 356 348 422 465 467 404 305 336 340
 362 363 435 491 505 359 310 337 360 342 406 396 420 472 548 559 463 407
 417 391 419 461 535 622 606 508 390 432]
```

```
df["month"].value_counts() #returns the count value of a columns
```

month	count
Jan	12
Feb	12
Mar	12
Apr	12
May	12
Jun	12
Jul	12
Aug	12
Sep	12
Oct	12
Nov	12
Dec	12

dtype: int64

```
df["month"].value_counts().reset_index()
```

	month	count
0	Jan	12
1	Feb	12
2	Mar	12
3	Apr	12
4	May	12
5	Jun	12
6	Jul	12
7	Aug	12
8	Sep	12
9	Oct	12
10	Nov	12
11	Dec	12

```
df.dropna() #delete that particular row if found any NaN value in any column of a row
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

```
df.drop_duplicates() #delete the duplicate rows in which all col has same value
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

```
df.drop_duplicates(subset=["month"]) #subset means delete on the basis of any particular col
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
5	1949	Jun	135
6	1949	Jul	148
7	1949	Aug	148
8	1949	Sep	136
9	1949	Oct	119
10	1949	Nov	104
11	1949	Dec	118

```
df.drop_duplicates(subset=["month"],keep="last") #if keep will be last, it stores the last occurance of the value
```

	year	month	passengers
132	1960	Jan	417
133	1960	Feb	391
134	1960	Mar	419
135	1960	Apr	461
136	1960	May	472
137	1960	Jun	535
138	1960	Jul	622
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

```
df.drop(columns=["month","year"], index=[2,3],errors="ignore") #columns means it will delete the complete col
#index means it will delete that particular index, ignore means ignore the error if any data not found
```

	passengers
0	112
1	118
4	121
5	135
6	148
...	...
139	606
140	508
141	461
142	390
143	432

142 rows × 1 columns

```
df_1 = df.iloc[0:, 0:2]
df_1
```

	year	month
0	1949	Jan
1	1949	Feb
2	1949	Mar
3	1949	Apr
4	1949	May
...	...	...
139	1960	Aug
140	1960	Sep
141	1960	Oct
142	1960	Nov
143	1960	Dec

144 rows × 2 columns

Next steps: Generate code with df_1 View recommended plots New interactive sheet

```
df_2 = df.iloc[0:4, 0:2]
df_2
```

	year	month
0	1949	Jan
1	1949	Feb
2	1949	Mar
3	1949	Apr

Next steps: Generate code with df_2 View recommended plots New interactive sheet

```
df.sort_index() #sort on the basis of index in the dataframe
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

```
df.sort_values(by="passengers") #sort on the basis of passengers in the dataframe
```

	year	month	passengers
10	1949	Nov	104
0	1949	Jan	112
22	1950	Nov	114
12	1950	Jan	115
11	1949	Dec	118
...	...	...	...
137	1960	Jun	535
126	1959	Jul	548
127	1959	Aug	559
139	1960	Aug	606
138	1960	Jul	622

144 rows × 3 columns

```
df.sort_values(by="passengers").reset_index() #given the index to the sorted dataframe
```

	index	year	month	passengers
0	10	1949	Nov	104
1	0	1949	Jan	112
2	22	1950	Nov	114
3	12	1950	Jan	115
4	11	1949	Dec	118
...	...	...	...	...
139	137	1960	Jun	535
140	126	1959	Jul	548
141	127	1959	Aug	559
142	139	1960	Aug	606
143	138	1960	Jul	622

144 rows × 4 columns

```
df.sort_values(by="passengers",ascending=False) #sort in descending order
```

	year	month	passengers
138	1960	Jul	622
139	1960	Aug	606
127	1959	Aug	559
126	1959	Jul	548
137	1960	Jun	535
...	...	...	...
11	1949	Dec	118
12	1950	Jan	115
22	1950	Nov	114
0	1949	Jan	112
10	1949	Nov	104

144 rows × 3 columns

```
import pandas as pd
My_dict = {
    "Name" : ["Gaurav", "Sourav", "Sachin", "Mohit"],
    "values" : [20,20,30,30],
}
my_df = pd.DataFrame(My_dict)
my_df
```

	Name	values
0	Gaurav	20
1	Sourav	20
2	Sachin	30
3	Mohit	30

Next steps: Generate code with my_df | View recommended plots | New interactive sheet

```
my_df.sort_values(by=["Name", "values"],ascending=False) #first sort on the basis of name then values in descendi
```

	Name	values
1	Sourav	20
2	Sachin	30
3	Mohit	30
0	Gaurav	20

```
df.set_index("month") #use month as a index not by default 0,1,2,3....
```

month	year	passengers
Jan	1949	112
Feb	1949	118
Mar	1949	132
Apr	1949	129
May	1949	121
...	...	...
Aug	1960	606
Sep	1960	508
Oct	1960	461
Nov	1960	390
Dec	1960	432

144 rows × 2 columns

```
df.set_index("year").loc[1949] #loc means location,give all values equal to 1949
```

year	month	passengers
1949	Jan	112
1949	Feb	118
1949	Mar	132
1949	Apr	129
1949	May	121
1949	Jun	135
1949	Jul	148
1949	Aug	148
1949	Sep	136
1949	Oct	119
1949	Nov	104
1949	Dec	118

```
import numpy as np
My_dict = {
    "Name" : ["Gaurav", "Sourav", "Sachin", "Mohit","rahul"],
    "values" : [20,20,30,30,np.nan],
}
my_df_2 = pd.DataFrame(My_dict)
my_df_2
```

	Name	values
0	Gaurav	20.0
1	Sourav	20.0
2	Sachin	30.0
3	Mohit	30.0
4	rahul	NaN

Next steps: Generate code with my_df_2 View recommended plots New interactive sheet

```
import warnings
my_df_2.fillna(my_df_2['values'].mean()).filter() #fill tha NaN value in values with mean of values
```

	Name	values
0	Gaurav	20.0
1	Sourav	20.0
2	Sachin	30.0
3	Mohit	30.0
4	rahul	25.0

```
import numpy as np
My_dict = {
    "Name" : ["Gaurav", np.nan, "Sachin", "Mohit","rahul"],
    "values" : [20,20,30,30,np.nan],
}
my_df_3 = pd.DataFrame(My_dict)
my_df_3
```

	Name	values
0	Gaurav	20.0
1	NaN	20.0
2	Sachin	30.0
3	Mohit	30.0
4	rahul	NaN

Next steps: Generate code with my_df_3 View recommended plots New interactive sheet

```
my_df_3["Name"].fillna(method="bfill") #fill tha NaN value with the forward or backward value of the dataframe
```

```
/tmp/ipython-input-45-3167234103.py:1: FutureWarning: Series.fillna with 'method' is deprecated and will rais
  my_df_3["Name"].fillna(method="bfill") #fill tha NaN value with mean
```

	Name
0	Gaurav
1	Sachin
2	Sachin
3	Mohit
4	rahul

dtype: object

```
df
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
def percentage(x) :
  return (x/622) * 100
percentage(134)
df["passengers_percent"] = percentage(df["passengers"]) #make a new col passengers oercentage and call percentage
df
```

	year	month	passengers	passengers_percent
0	1949	Jan	112	18.006431
1	1949	Feb	118	18.971061
2	1949	Mar	132	21.221865
3	1949	Apr	129	20.739550
4	1949	May	121	19.453376
...	...	...	...	...
139	1960	Aug	606	97.427653
140	1960	Sep	508	81.672026
141	1960	Oct	461	74.115756
142	1960	Nov	390	62.700965
143	1960	Dec	432	69.453376

144 rows × 4 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df["p_Percentage"] = df["passengers"].apply(percentage) #apply any external function
df
```

	year	month	passengers	passengers_percent	p_Percentage
0	1949	Jan	112	18.006431	18.006431
1	1949	Feb	118	18.971061	18.971061
2	1949	Mar	132	21.221865	21.221865
3	1949	Apr	129	20.739550	20.739550
4	1949	May	121	19.453376	19.453376
...	...	...	...	...	...
139	1960	Aug	606	97.427653	97.427653
140	1960	Sep	508	81.672026	81.672026
141	1960	Oct	461	74.115756	74.115756
142	1960	Nov	390	62.700965	62.700965
143	1960	Dec	432	69.453376	69.453376

144 rows × 5 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df["month"].apply(len) #len of the string of month
```

	month
0	3
1	3
2	3
3	3
4	3
...	...
139	3
140	3
141	3
142	3
143	3

144 rows × 1 columns

dtype: int64

Group by function in pandas

```
df.head(20)
```

	year	month	passengers	passengers_percent	p_Percentage
0	1949	Jan	112	18.006431	18.006431