Nama: Muhammad Haidar Abdul Jabbar

Kelas : TK-44-G6

Nim:1103202071

Tugas II

Pertemuan 5-V(2)

1. Install Modul kaggle:

```
# Install modul kaggle secara inline (di dalam notebook)
!pip install kaggle
```

✓ 1.5s Python

```
Requirement already satisfied: kaggle in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (1.5.16)
Requirement already satisfied: six≥1.10 in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (2023.7.22)
Requirement already satisfied: python-dateutil in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (2.8.2
Requirement already satisfied: requests in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (4.66.1)
Requirement already satisfied: python-slugify in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (2.0.6)
Requirement already satisfied: bleach in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from bleach→kaggle) (
Requirement already satisfied: text-unidecode≥1.3 in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from python-slu
Requirement already satisfied: charset-normalizer<4,≥2 in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from reque
Requirement already satisfied: idna<4,≥2.5 in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from requests→kaggle)
Requirement already satisfied: colorama in c:\users\muham\appdata\local\programs\python\python312\lib\site-packages (from tqdm→kaggle) (0.4.6)
```

Mengistall kaggle pada local computer -> membuat api key -> menaruh json key pada local komputer

Tutorial lengkap: https://github.com/Kaggle/kaggle-api

Melakukan list dataset yang tersedia berdasarkan keyword pada Kaggle

```
# Mencari dataset yang tersedia di kaggle ⟶ pilih dataset iris dengan provider dari UCIML
!kaggle datasets list -s "iris"
```

✓ 1.8s Python

ref	title	size	lastUpdated	downlo
uciml/iris	Iris Species	4KB	2016-09-27 07:38:05	
arshid/iris-flower-dataset	Iris Flower Dataset	1010B	2018-03-22 15:18:06	
himanshunakrani/iris-dataset	Iris dataset	1006B	2022-07-20 18:50:06	
vikrishnan/iris-dataset	Iris Dataset	999B	2017-08-03 16:00:44	
parulpandey/palmer-archipelago-antarctica-penguin-data	Palmer Archipelago (Antarctica) penguin data	11KB	2020-06-09 10:14:54	
chuckyin/iris-datasets	Iris datasets	1KB	2017-03-10 09:35:43	
jillanisofttech/iris-dataset-uci	Iris dataset uci	1KB	2021-11-06 15:11:47	
rtatman/iris-dataset-json-version	Iris Dataset (JSON Version)	1KB	2018-04-06 20:21:31	
therohk/ireland-historical-news	Irish Times - Waxy-Wany News	52MB	2021-09-25 10:52:48	
vijayaadithyanvg/iris-dataset	IRIS DATASET	1023B	2023-01-14 07:44:53	
arslanali4343/iris-species	Iris Species	2KB	2020-07-02 06:09:09	
jeffheaton/iris-computer-vision	Iris Computer Vision	5MB	2020-11-24 21:23:29	
saurabh00007/iriscsv	Iris.csv	1KB	2017-11-09 07:34:35	
fleanend/birds-songs-numeric-dataset	Birds' Songs Numeric Dataset	25MB	2019-04-01 09:09:46	
conorrot/irish-weather-hourly-data	Irish Weather (hourly data)	67MB	2020-06-29 20:15:18	
arunjangir245/irisraw	irisraw	1017B	2023-07-31 21:02:17	
sims22/irisflowerdatasets	IRIS-FLOWER-DATASETS	2KB	2022-12-26 18:33:34	
naureenmohammad/mmu-iris-dataset	MMU iris dataset	30MB	2020-07-25 18:38:33	
ashkhagan/palmer-penguins-datasetalternative-iris-dataset	Palmer Penguins Dataset-Alternative Iris Dataset	3KB	2020-06-30 06:30:43	
kamrankausar/iris-data	iris_data	1KB	2017-11-30 10:26:01	

Melakukan download dan unzip dataset

```
# Download dan ekstrak dataset, secara default akan berada dalam satu direktori dengan notebook ini
!kaggle datasets download uciml/iris --unzip
```

[3] ✓ 2.2s

```
Downloading iris.zip to c:\allDataSaya\kampus\Telkom University\tugas\tingkat 4\data sains\02_Tugas\p5

  0%|          | 0.00/3.60k [00:00<?, ?B/s]
100%|██████████| 3.60k/3.60k [00:00<00:00, 3.70MB/s]
```

Membuat dataset menjadi dataframe dan menunjukan 10 row pertama

```
# import library
import pandas as pd
import numpy as np
```

Python

```
# load dataset
dataset = "epl-goalScorer(20-21).csv"
df = pd.read_csv(dataset)
```

Python

```
# menampilkan 5 baris awal dari dataset
df.head(10)
```

Python

	Unnamed: 0	id	player_name	games	time	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	position	team_title	npg	npxG	
0	0	647	Harry Kane	35	3097	23	22.174859	14	7.577094	138	49	1	0	F	Tottenham	19	19.130183	2
1	1	1250	Mohamed Salah	37	3085	22	20.250847	5	6.528526	126	55	0	0	F M S	Liverpool	16	15.683834	2
2	2	1228	Bruno Fernandes	37	3117	18	16.019454	12	11.474996	121	95	6	0	M S	Manchester United	9	8.407840	2
3	3	453	Son Heung-Min	37	3139	17	11.023287	10	9.512992	68	75	0	0	F M S	Tottenham	16	10.262118	2
4	4	822	Patrick Bamford	38	3085	17	18.401863	7	3.782247	107	30	3	0	F S	Leeds	15	16.879525	2
5	5	5555	Dominic Calvert-	32	2788	16	18.136449	0	2.101819	82	18	3	0	F S	Everton	16	18.136449	1

Menunjukan 7 row terakhir dataframe

```
# menampilkan 5 baris akhir dari dataset
df.tail(7)
```

Python

	Unnamed: 0	id	player_name	games	time	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	position	team_title	npg	npxG	xGChain
515	515	9395	Sidnei Tavares	2	84	0	0.020798	0	0.0	1	0	0	0	M S	Leicester	0	0.020798	0.02079
516	516	9406	Nathan Broadhead	1	1	0	0.000000	0	0.0	0	0	0	0	S	Everton	0	0.000000	0.00000
517	517	9415	Jaden Philogene-Bidace	1	1	0	0.000000	0	0.0	0	0	0	0	S	Aston Villa	0	0.000000	0.05604
518	518	9423	Gaetano Berardi	2	113	0	0.074761	0	0.0	1	0	0	0	D S	Leeds	0	0.074761	0.23127
519	519	9524	Anthony Elanga	1	67	0	0.000000	0	0.0	0	0	0	0	M	Manchester United	0	0.000000	0.00000
520	520	9540	Femi Seriki	1	1	0	0.000000	0	0.0	0	0	0	0	S	Sheffield United	0	0.000000	0.00000
521	521	9552	Tyrese Francois	1	13	0	0.000000	0	0.0	0	0	0	0	S	Fulham	0	0.000000	0.00000

Bertujuan untuk mengetahui type data setiap column

```
# mengungkap tipe-tipe data dari setiap kolom
print(df.dtypes)
```

```
Unnamed: 0         int64
id                 int64
player_name       object
games              int64
time               int64
goals              int64
xG               float64
assists            int64
xA               float64
shots              int64
key_passes         int64
yellow_cards       int64
red_cards          int64
position          object
team_title        object
npg                int64
npxG             float64
xGChain          float64
xGBuildup        float64
dtype: object
```

Melakukan slicing row & column [:,2:]

#maksud [row,column]

#[semua row,mulai column 2 sampai terkhir]

```
df_noid = df.iloc[:,2:]
df_noid
```

	player_name	games	time	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	position	team_title	npg	npxG	xGChain	xGBuildup
0	Harry Kane	35	3097	23	22.174859	14	7.577094	138	49	1	0	F	Tottenham	19	19.130183	24.995648	4.451257
1	Mohamed Salah	37	3085	22	20.250847	5	6.528526	126	55	0	0	F M S	Liverpool	16	15.683834	28.968234	9.800236
2	Bruno Fernandes	37	3117	18	16.019454	12	11.474996	121	95	6	0	M S	Manchester United	9	8.407840	26.911412	11.932285
3	Son Heung-Min	37	3139	17	11.023287	10	9.512992	68	75	0	0	F M S	Tottenham	16	10.262118	20.671916	6.608751
4	Patrick Bamford	38	3085	17	18.401863	7	3.782247	107	30	3	0	F S	Leeds	15	16.879525	23.394953	4.131796
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
517	Jaden Philogene-Bidace	1	1	0	0.000000	0	0.000000	0	0	0	0	S	Aston Villa	0	0.000000	0.056044	0.056044
518	Gaetano Berardi	2	113	0	0.074761	0	0.000000	1	0	0	0	D S	Leeds	0	0.074761	0.231278	0.231278
519	Anthony Elanga	1	67	0	0.000000	0	0.000000	0	0	0	0	M	Manchester United	0	0.000000	0.000000	0.000000
520	Femi Seriki	1	1	0	0.000000	0	0.000000	0	0	0	0	S	Sheffield United	0	0.000000	0.000000	0.000000
521	Tyrese Francois	1	13	0	0.000000	0	0.000000	0	0	0	0	S	Fulham	0	0.000000	0.000000	0.000000

522 rows × 17 columns

Menampilkan statistic dasar

```
# menampilkan statistik dasar setiap kolom data yang bertipe numerik.
df_noid.describe()
```

	games	time	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	npg	npxG	xGChain	xGBuildup
count	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000	522.000000
mean	19.643678	1420.068966	1.862069	2.000806	1.289272	1.376029	17.379310	12.963602	2.061303	0.091954	1.668582	1.821450	5.663368	3.455060
std	11.619836	1031.604819	3.338851	3.317946	2.083350	1.886510	21.572664	16.164361	2.203661	0.295800	2.909929	2.931176	5.600249	3.376584
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	10.000000	470.250000	0.000000	0.074668	0.000000	0.049245	2.000000	1.000000	0.000000	0.000000	0.000000	0.074668	1.191391	0.720353
50%	21.000000	1342.000000	1.000000	0.737295	0.000000	0.691122	10.000000	7.000000	2.000000	0.000000	0.500000	0.715585	4.252738	2.656397
75%	30.000000	2319.000000	2.000000	2.053378	2.000000	2.050509	23.750000	19.000000	3.000000	0.000000	2.000000	1.945799	8.308002	5.254647
max	38.000000	3420.000000	23.000000	22.174859	14.000000	11.474996	138.000000	95.000000	12.000000	2.000000	19.000000	19.130183	28.968234	18.323006

Menunjukan jumlah sum() dan rata2 mean() dataframe

```
df_noid.mean()
```

```
games            19.643678
time           1420.068966
goals             1.862069
xG                2.000806
assists           1.289272
xA                1.376029
shots            17.379310
key_passes       12.963602
yellow_cards      2.061303
red_cards         0.091954
npg               1.668582
npxG              1.821450
xGChain           5.663368
xGBuildup         3.455060
dtype: float64
```

```
df_noid.sum()
```

```
player_name     Harry KaneMohamed SalahBruno FernandesSon Heun ...
games                                                       10254
time                                                       741276
goals                                                         972
xG                                                    1044.420572
assists                                                       673
xA                                                     718.287269
shots                                                        9072
key_passes                                                   6767
yellow_cards                                                 1076
red_cards                                                      48
position        FF M SM SF M SF SF SF SFM SF M SF SF SF SF SF  ...
team_title      TottenhamLiverpoolManchester UnitedTottenhamLe ...
npg                                                           871
npxG                                                    950.7971
xGChain                                                2956.278233
xGBuildup                                              1803.541131
```

Menunjukan nilai Tengah dataframe median() dan variance var()

```
df_noid.median()
```

```
games             21.000000
time            1342.000000
goals              1.000000
xG                 0.737295
assists            0.000000
xA                 0.691122
shots             10.000000
key_passes         7.000000
yellow_cards       2.000000
red_cards          0.000000
npg                0.500000
npxG               0.715585
xGChain            4.252738
xGBuildup          2.656397
dtype: float64
```

```
df_noid.var()
```

```
games           1.350206e+02
time            1.064209e+06
goals           1.114793e+01
xG              1.100877e+01
assists         4.340345e+00
xA              3.558919e+00
shots           4.653798e+02
key_passes      2.612866e+02
yellow_cards    4.856120e+00
red_cards       8.749752e-02
npg             8.467687e+00
npxG            8.591795e+00
xGChain         3.136279e+01
xGBuildup       1.140132e+01
dtype: float64
```

Menunjukan standard deviasion std() dan quartil 3

```
df_noid.std()
```

```
games             11.619836
time            1031.604819
goals              3.338851
xG                 3.317946
assists            2.083350
xA                 1.886510
shots             21.572664
key_passes        16.164361
yellow_cards       2.203661
red_cards          0.295800
npg                2.909929
npxG               2.931176
xGChain            5.600249
xGBuildup          3.376584
dtype: float64
```

```
df_noid.quantile(0.75)
```

```
games             30.000000
time            2319.000000
goals              2.000000
xG                 2.053378
assists            2.000000
xA                 2.050509
shots             23.750000
key_passes        19.000000
yellow_cards       3.000000
red_cards          0.000000
npg                2.000000
npxG               1.945799
xGChain            8.308002
xGBuildup          5.254647
Name: 0.75, dtype: float64
```

mencari nilai range

Mencari pencilan dengan Tukey's fences (1)

```
q1 = df_noid.quantile(0.25)
q3 = df_noid.quantile(0.75)
iqr = q3 - q1
iqr
```

```
games             20.000000
time            1848.750000
goals              2.000000
xG                 1.978711
assists            2.000000
xA                 2.001264
shots             21.750000
key_passes        18.000000
yellow_cards       3.000000
red_cards          0.000000
npg                2.000000
npxG               1.871131
xGChain            7.116612
xGBuildup          4.534294
dtype: float64
```

Mencari nilai outlier

```
(df_noid < q1 - 1.5 * iqr_new) | (df_noid > q3 + 1.5 * iqr_new)
```

Mencari pencilan dengan Tukey's fences (2)

```
# handle warning
import warnings
warnings.filterwarnings('ignore')

# outlier filter
df_noid_align, iqr_new = df_noid.align(iqr, axis=1, copy=False, join='outer')
outlier_filter = (df_noid < q1 - 1.5 * iqr_new) | (df_noid > q3 + 1.5 * iqr_new)
outlier_filter
```

Melakukan slicing column player dan assist serta sorting ascending berdasarkan asist

```
df_noid[outlier_filter['assists']] \
    .loc[:, ['player_name', 'assists']] \
    .sort_values(by=['assists'], ascending=Fal
```

	player_name	assists
0	Harry Kane	14
2	Bruno Fernandes	12
58	Kevin De Bruyne	11
3	Son Heung-Min	10
51	Jack Grealish	10
6	Jamie Vardy	9
15	Marcus Rashford	9
57	Raphinha	9
41	Jack Harrison	8
281	Aaron Cresswell	8
83	Pascal Groß	8
49	Timo Werner	8
32	James Ward-Prowse	7
26	Roberto Firmino	7
16	Sadio Mané	7
130	Trent Alexander-Arnold	7
204	Andrew Robertson	7
4	Patrick Bamford	7
358	Lucas Digne	7
42	Bertrand Traoré	6

Menghitung total nilai berdasarkan column team title dan time

Value_counts()

menghasilkan frekuensi setiap nilai unik di dalam kolom, yang tertinggi count-ny
dalam musim yang sama(ada transfer pemain)

```
[92]  df_noid[['team_title','time']].value_counts()
```

```
team_title              time
Leicester               17      2
Brighton                201     2
Tottenham               3420    2
Manchester United       85      2
Manchester City         90      2
                                ..
Everton                 1612    1
                        1574    1
                        1505    1
                        1343    1
Wolverhampton Wanderers 3240    1
Length: 515, dtype: int64
```

Melakukan pivot standard deviasion column team title berdasarkan goals

Analisis dengan groupby

method groupby memungkinkan analisa dilakuka

```
df.groupby('team_title')['goals'].std(
```

```
team_title
Arsenal                          3.352381
Arsenal,Brighton                      NaN
Arsenal,Newcastle United              NaN
Arsenal,West Bromwich Albion          NaN
Aston Villa                      3.696489
Aston Villa,Chelsea                   NaN
Brighton                         2.158703
Burnley                          2.475210
Chelsea                          2.350177
Chelsea,Fulham                        NaN
Crystal Palace                   2.901461
Everton                          3.467727
Everton,Southampton                   NaN
Fulham                           1.439175
Leeds                            4.153193
Leicester                        4.020602
Liverpool                        4.931439
Liverpool,Southampton                 NaN
Manchester City                  3.867132
Manchester United                4.317855
Newcastle United                 2.483174
Sheffield United                 1.467599
Southampton                      3.141941
Tottenham                        5.855135
West Bromwich Albion             2.310260
West Bromwich Albion,West Ham         NaN
West Ham                         3.369240
Wolverhampton Wanderers          1.648620
Name: goals, dtype: float64
```

Melakukan pivot rata2 column team title berdasarkan goals

```
df.groupby('team_title')['goals'].mean()
```

```
team_title
Arsenal                          1.961538
Arsenal,Brighton                 0.000000
Arsenal,Newcastle United         8.000000
Arsenal,West Bromwich Albion     0.000000
Aston Villa                      2.130435
Aston Villa,Chelsea              3.000000
Brighton                         1.500000
Burnley                          1.280000
Chelsea                          2.240000
Chelsea,Fulham                   1.000000
Crystal Palace                   1.625000
Everton                          1.607143
Everton,Southampton              3.000000
Fulham                           0.925926
Leeds                            2.608696
Leicester                        2.370370
Liverpool                        2.370370
Liverpool,Southampton            3.000000
Manchester City                  3.208333
Manchester United                2.518519
Newcastle United                 1.384615
Sheffield United                 0.666667
Southampton                      1.555556
Tottenham                        2.750000
West Bromwich Albion             1.178571
West Bromwich Albion,West Ham    0.000000
West Ham                         2.478261
Wolverhampton Wanderers          1.222222
Name: goals, dtype: float64
```

Menunjukan corelasi nilai column mulai column games dengan seterusnya

Korelasi Pearson antara kolom-kolom numerik

- Method corr() menghasilkan tabel korelasi pearson antar kolom-kolom numerik.
- Rentang nilai: antara -1 dan 1
- -1 = korelasi negatif | 0 = tidak ada korelasi linier | +1 = korelasi positif

```
df_noid.loc[:,'games':].corr()
```

	games	time	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	npg	npxG	xGChain	xGBuildup
games	1.000000	0.944591	0.439730	0.463869	0.504168	0.562806	0.599164	0.617867	0.565963	0.160326	0.437110	0.465546	0.726598	0.697196
time	0.944591	1.000000	0.398930	0.411203	0.473555	0.516638	0.529534	0.575065	0.592223	0.186333	0.392631	0.408231	0.703801	0.731377
goals	0.439730	0.398930	1.000000	0.932798	0.617490	0.607330	0.873363	0.567752	0.097151	0.053679	0.971591	0.905710	0.727953	0.290990
xG	0.463869	0.411203	0.932798	1.000000	0.636205	0.627495	0.910214	0.570488	0.093761	0.048815	0.894286	0.979218	0.763909	0.282746
assists	0.504168	0.473555	0.617490	0.636205	1.000000	0.885850	0.721220	0.835299	0.209349	-0.021444	0.587316	0.615503	0.752587	0.473254
xA	0.562806	0.516638	0.607330	0.627495	0.885850	1.000000	0.759568	0.946506	0.243912	0.006284	0.585152	0.611100	0.814487	0.547983
shots	0.599164	0.529534	0.873363	0.910214	0.721220	0.759568	1.000000	0.743370	0.249957	0.073932	0.852989	0.901386	0.843152	0.448197
key_passes	0.617867	0.575065	0.567752	0.570488	0.835299	0.946506	0.743370	1.000000	0.343357	0.022780	0.539726	0.545537	0.807958	0.618754
yellow_cards	0.565963	0.592223	0.097151	0.093761	0.209349	0.243912	0.249957	0.343357	1.000000	0.165064	0.093270	0.089065	0.401884	0.562467
red_cards	0.160326	0.186333	0.053679	0.048815	-0.021444	0.006284	0.073932	0.022780	0.165064	1.000000	0.055542	0.047354	0.104005	0.167660
npg	0.437110	0.392631	0.971591	0.894286	0.587316	0.585152	0.852989	0.539726	0.093270	0.055542	1.000000	0.913496	0.720978	0.284135
npxG	0.465546	0.408231	0.905710	0.979218	0.615503	0.611100	0.901386	0.545537	0.089065	0.047354	0.913496	1.000000	0.763481	0.273090
xGChain	0.726598	0.703801	0.727953	0.763909	0.752587	0.814487	0.843152	0.807958	0.401884	0.104005	0.720978	0.763481	1.000000	0.802073
xGBuildup	0.697196	0.731377	0.290990	0.282746	0.473254	0.547983	0.448197	0.618754	0.562467	0.167660	0.284135	0.273090	0.802073	1.000000

Pertemuan 6

Melakukan import library

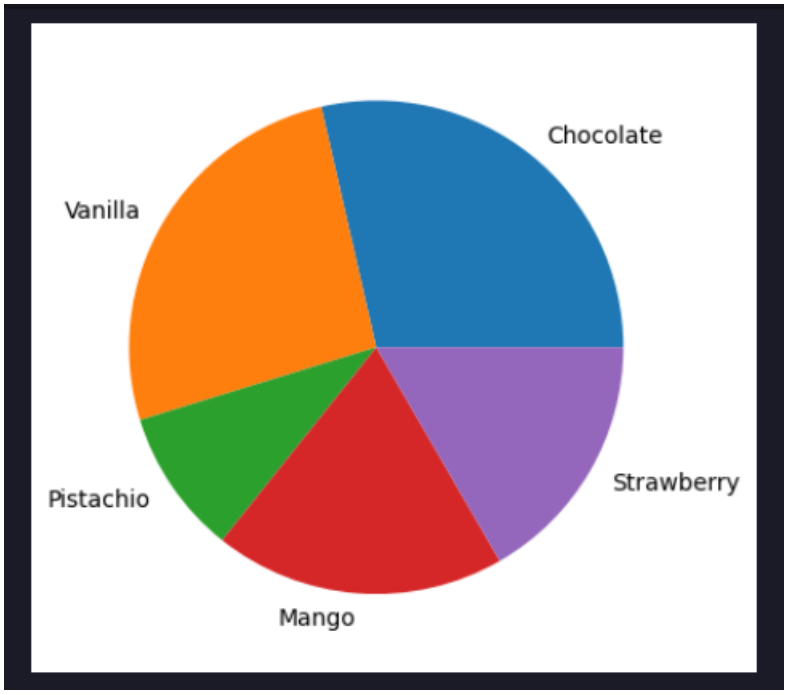
Visualisasi Variabel

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

[31] ✓ 0.0s

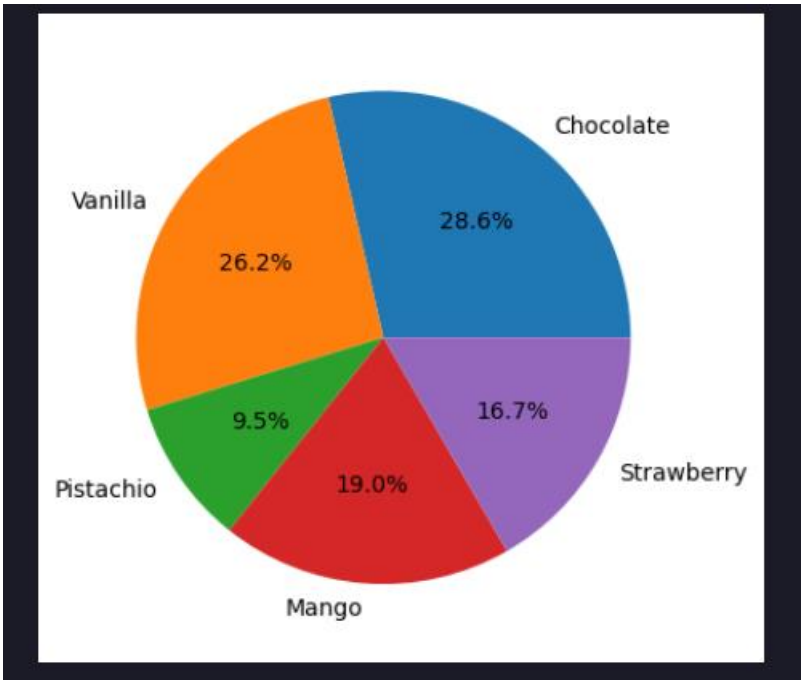
Visualisasi pie chart

```
# visualisasi pie chart
flavors = ('Chocolate', 'Vanilla', 'Pistachio', 'Mango', 'Strawberry')
votes = (12, 11, 4, 8, 7)
plt.pie(
    votes,
    labels=flavors,
)
plt.show()
```



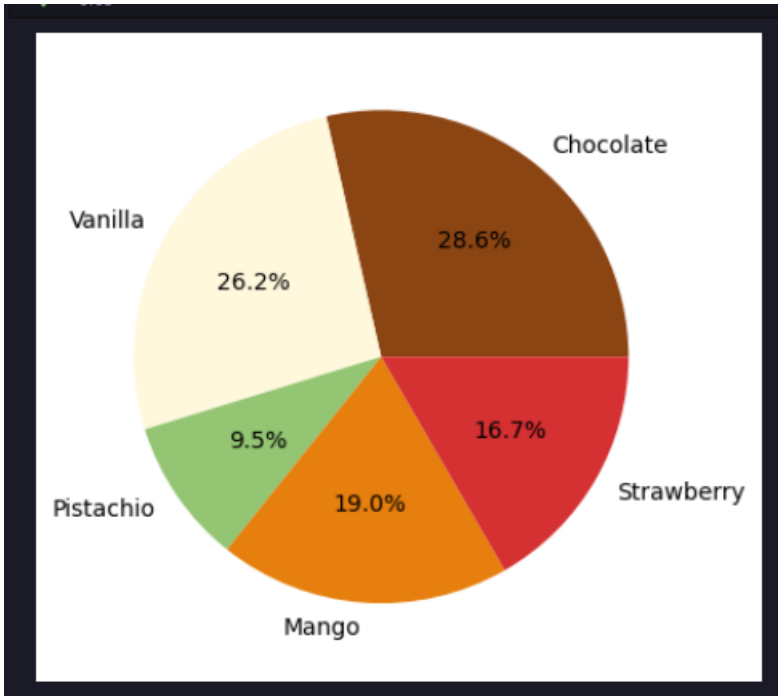
Visualisasi pie chart dengan persentase

```python
# melihat persen kontribusi pada pie chart
flavors = ('Chocolate', 'Vanilla', 'Pistachio', 'Mango', 'Strawberry')
votes = (12, 11, 4, 8, 7)
plt.pie(
    votes,
    labels=flavors,
    autopct='%1.1f%%',
)
plt.show()
```



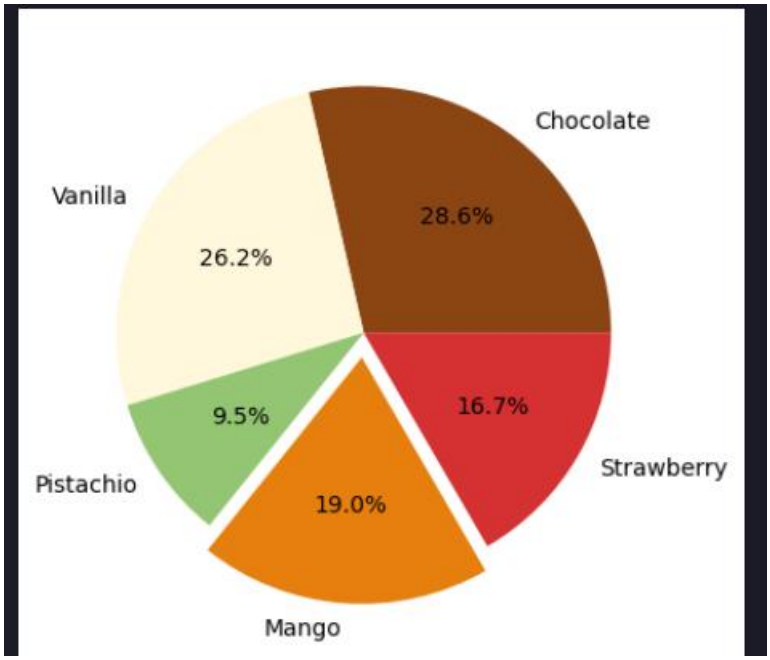
Kostom warna dengan pychart

```python
# kostumisasi warna pie chart
flavors = ('Chocolate', 'Vanilla', 'Pistachio', 'Mango', 'Strawberry')
votes = (12, 11, 4, 8, 7)
colors = ('#8B4513', '#FFF8DC', '#93C572', '#E67F0D', '#D53032')
plt.pie(
    votes,
    labels=flavors,
    autopct='%1.1f%%',
    colors=colors,
)
plt.show()
```



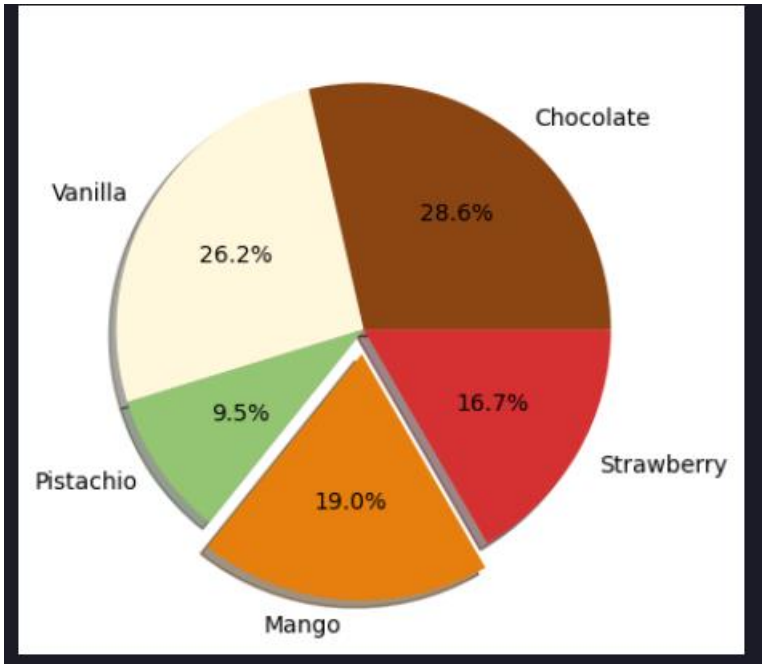
Menggunakan explode sebagai data terdapat potongan

```
# menambah highlight terhadap data
flavors = ('Chocolate', 'Vanilla', 'Pistachio', 'Mango', 'Strawberry')
votes = (12, 11, 4, 8, 7)
colors = ('#8B4513', '#FFF8DC', '#93C572', '#E67F0D', '#D53032')
explode = (0, 0, 0, 0.1, 0)
plt.pie(
    votes,
    labels=flavors,
    autopct='%1.1f%%',
    colors=colors,
    explode=explode,
)
plt.show()
```



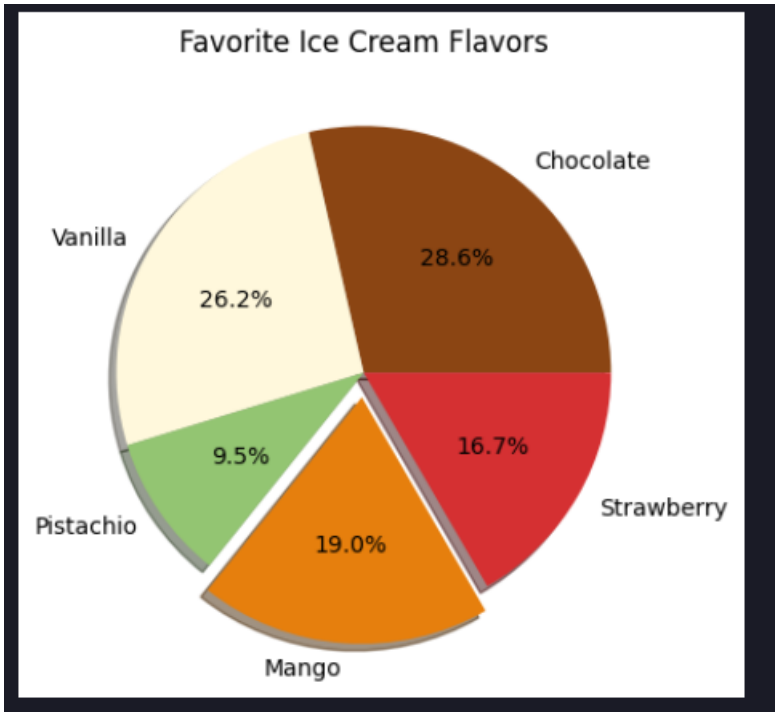
Shadow berfungsi untuk membuat pie chart terdapat bayangan

```
# menambah bayangan visualisasi
flavors = ('Chocolate', 'Vanilla', 'Pistachio', 'Mango', 'Strawberry')
votes = (12, 11, 4, 8, 7)
colors = ('#8B4513', '#FFF8DC', '#93C572', '#E67F0D', '#D53032')
explode = (0, 0, 0, 0.1, 0)
plt.pie(
    votes,
    labels=flavors,
    autopct='%1.1f%%',
    colors=colors,
    explode=explode,
    shadow=True
    )
plt.show()
```



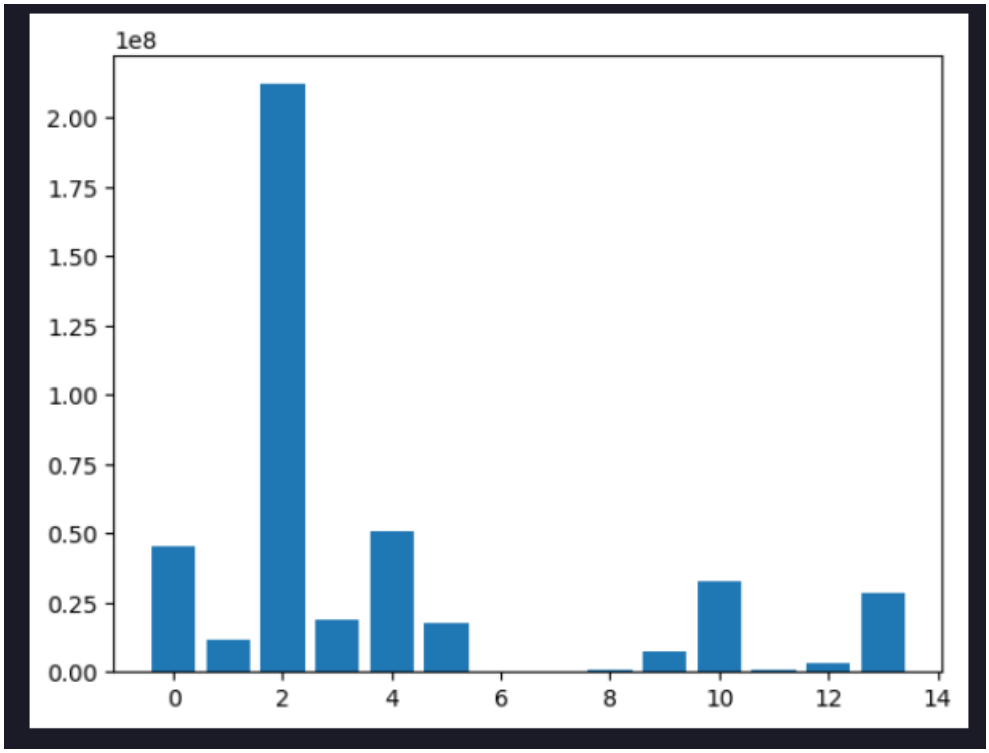
Menggunakan title sebagai judul pie chart

```
# menambahkan judul
flavors = ('Chocolate', 'Vanilla', 'Pistachio', 'Mango', 'Strawberry')
votes = (12, 11, 4, 8, 7)
colors = ('#8B4513', '#FFF8DC', '#93C572', '#E67F0D', '#D53032')
explode = (0, 0, 0, 0.1, 0)

plt.title('Favorite Ice Cream Flavors')
plt.pie(
    votes,
    labels=flavors,
    autopct='%1.1f%%',
    colors=colors,
    explode=explode,
    shadow=True
    )
plt.show()
```



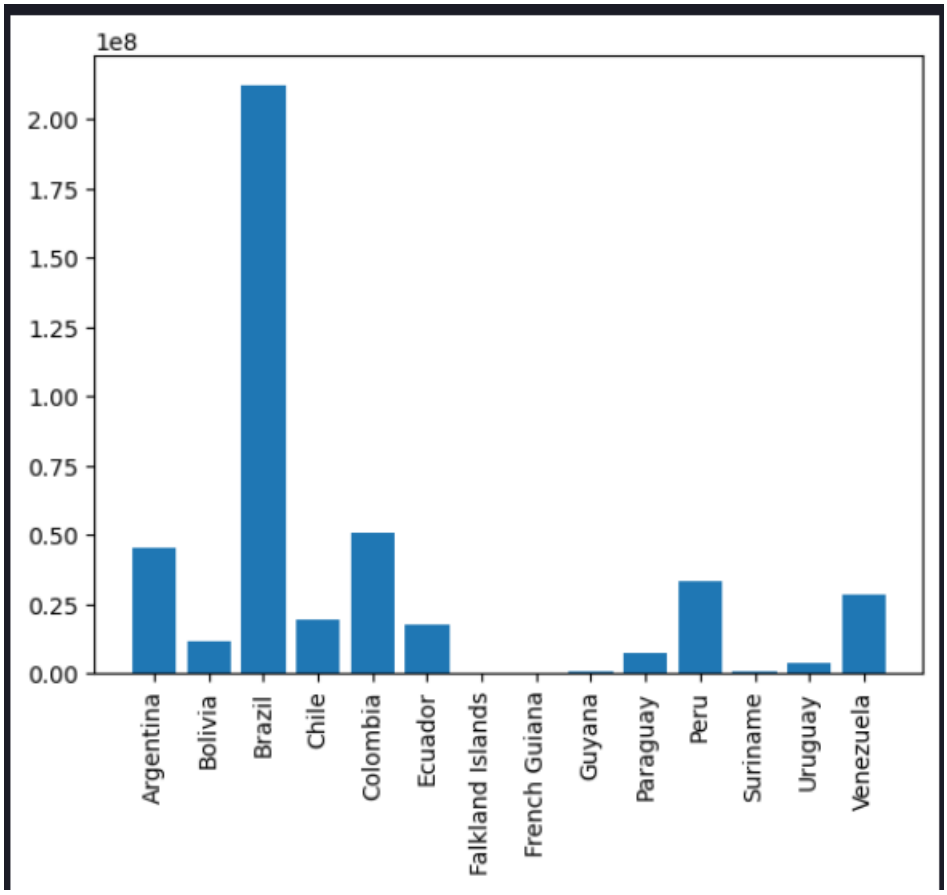
Membuat barchat

```
# memuat data
countries = ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador',
             'Falkland Islands', 'French Guiana', 'Guyana', 'Paraguay', 'Peru',
             'Suriname', 'Uruguay', 'Venezuela')

populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,
               3481, 287750, 785409, 7107305, 32880332, 585169, 3470475,
               28258770)

# visualisasi bar chart
x_coords = np.arange(len(countries))
plt.bar(x_coords, populations)
plt.show()
```



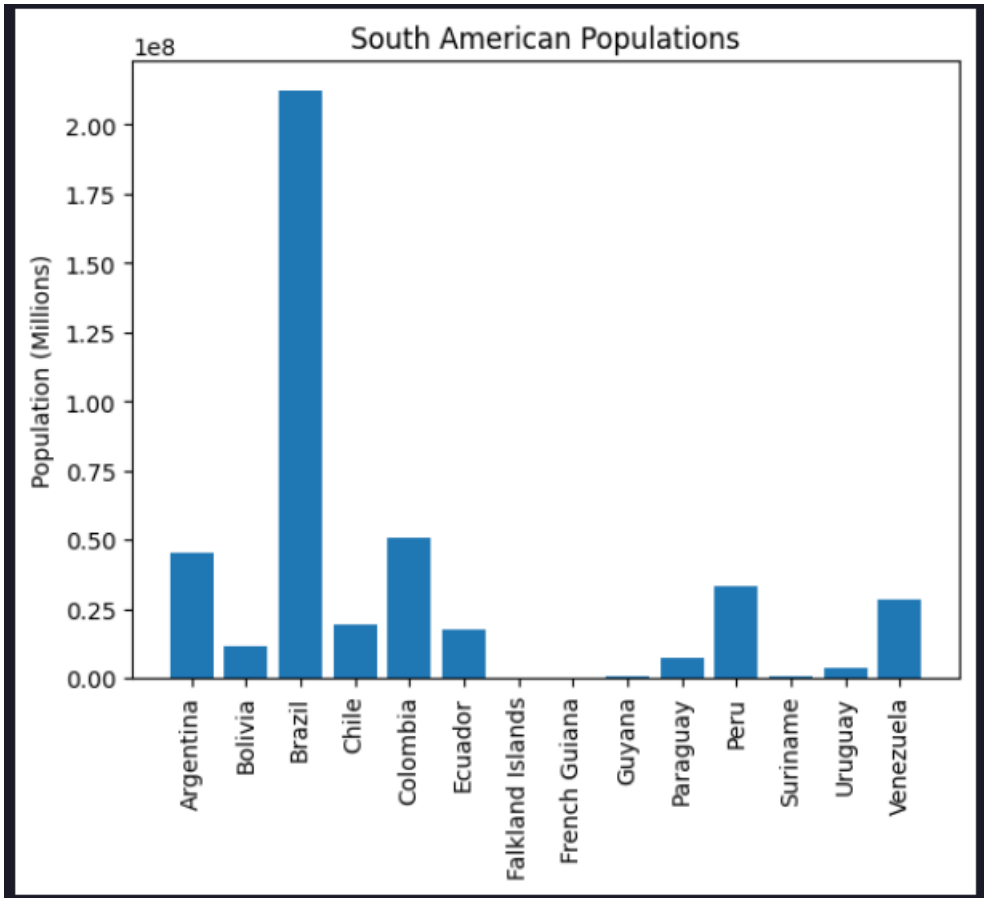
Memutar/rotasi pada label

```
# memutar label 90 derajat
countries = ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador',
             'Falkland Islands', 'French Guiana', 'Guyana', 'Paraguay', 'Peru',
             'Suriname', 'Uruguay', 'Venezuela')

populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,
               3481, 287750, 785409, 7107305, 32880332, 585169, 3470475,
               28258770)

x_coords = np.arange(len(countries))
plt.bar(x_coords, populations, tick_label=countries)
plt.xticks(rotation=90) #memutar labels x-axis
plt.show()
```



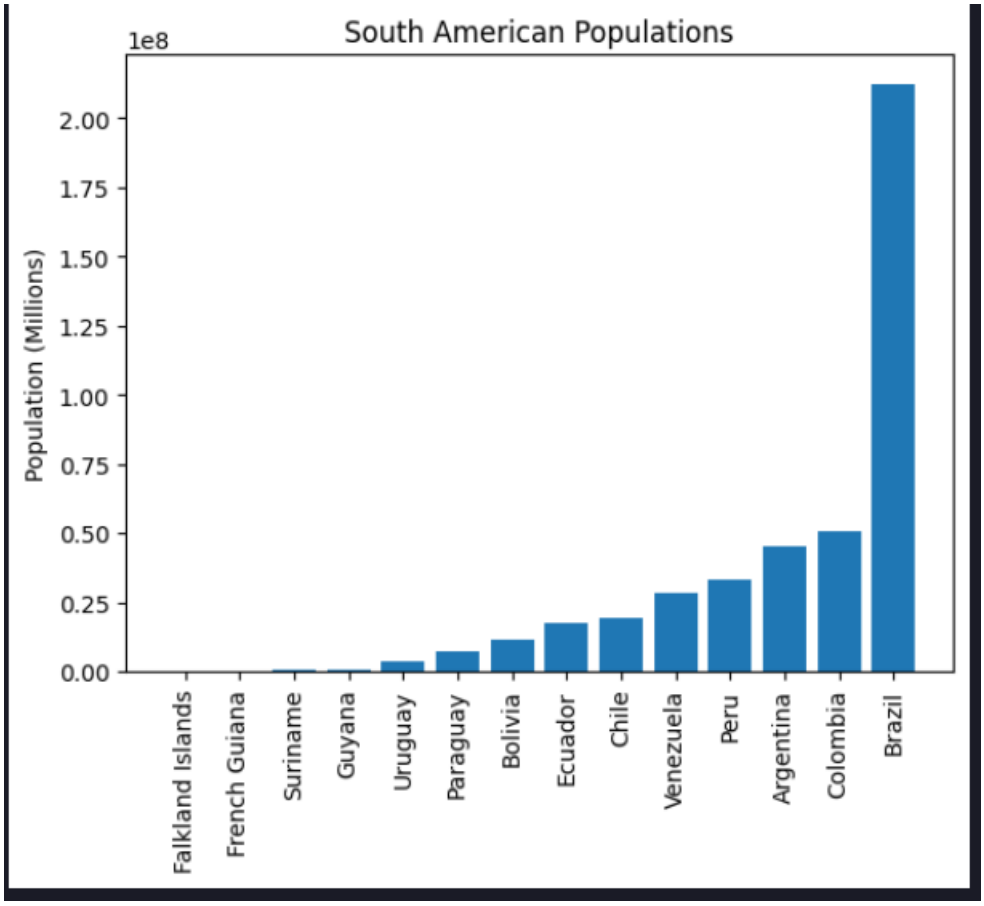
Membbuat label pada diagram

```
# menambah label y
countries = ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador',
             'Falkland Islands', 'French Guiana', 'Guyana', 'Paraguay', 'Peru',
             'Suriname', 'Uruguay', 'Venezuela')

populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,
               3481, 287750, 785409, 7107305, 32880332, 585169, 3470475,
               28258770)

x_coords = np.arange(len(countries))
plt.bar(x_coords, populations, tick_label=countries)
plt.xticks(rotation=90)
plt.ylabel('Population (Millions)')
plt.title('South American Populations')
plt.show()
```



Mengurutkan data pada diagram

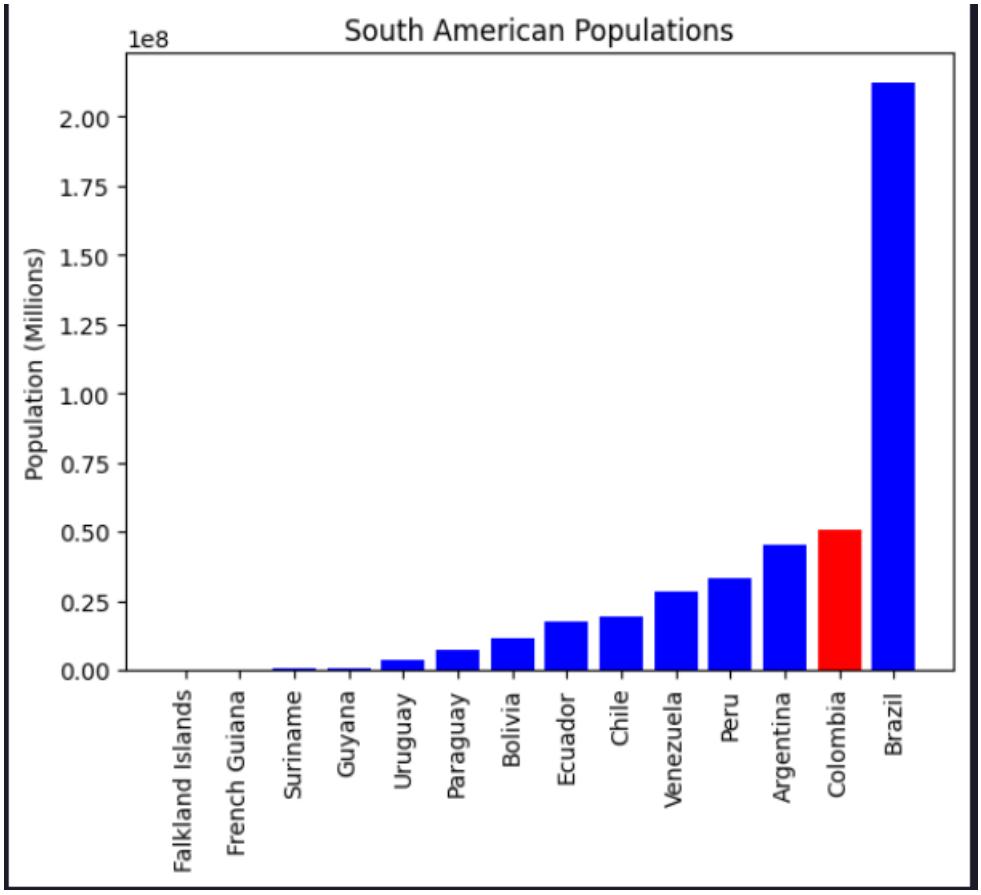
```
# mengurutkan data
countries = ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador',
             'Falkland Islands', 'French Guiana', 'Guyana', 'Paraguay', 'Peru',
             'Suriname', 'Uruguay', 'Venezuela')

populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,
               3481, 287750, 785409, 7107305, 32880332, 585169, 3470475,
               28258770)

df = pd.DataFrame({
    'Country': countries,
    'Population': populations,
})
df.sort_values(by='Population', inplace=True)

x_coords = np.arange(len(df))
plt.bar(x_coords, df['Population'], tick_label=df['Country'])
plt.xticks(rotation=90)
plt.ylabel('Population (Millions)')
plt.title('South American Populations')
plt.show()
```



Memberi warna spesifik pada table

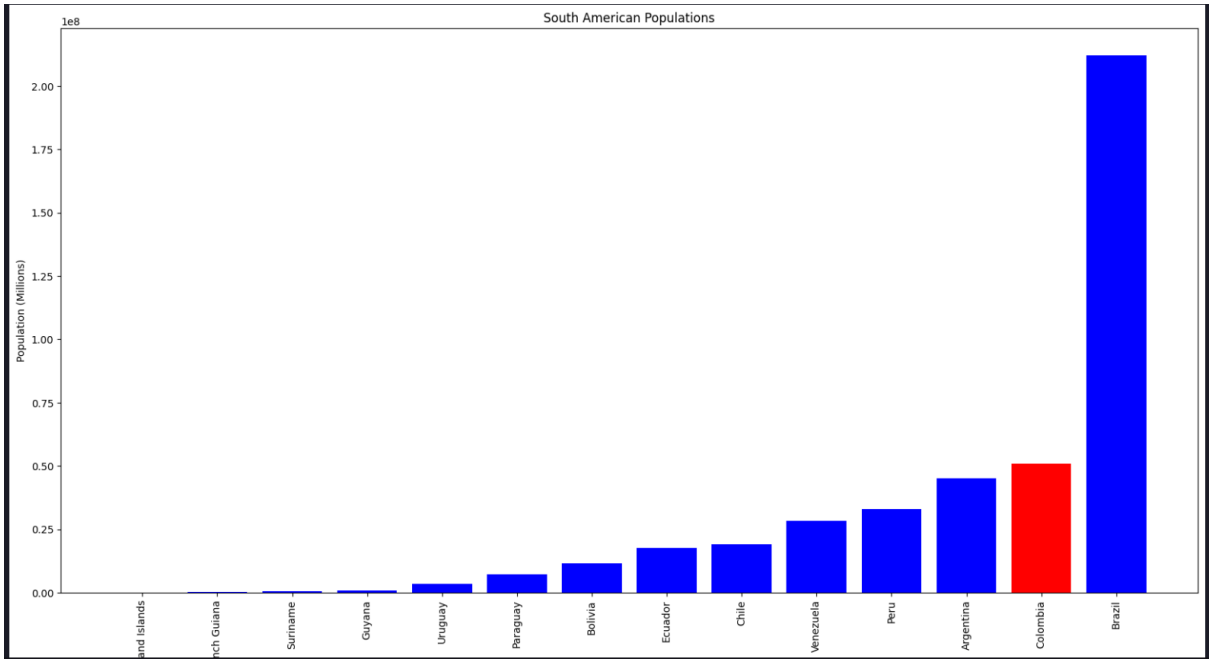
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

countries = ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador',
             'Falkland Islands', 'French Guiana', 'Guyana', 'Paraguay', 'Peru',
             'Suriname', 'Uruguay', 'Venezuela')

populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,
               3481, 287750, 785409, 7107305, 32880332, 585169, 3470475,
               28258770)

df = pd.DataFrame({
    'Country': countries,
    'Population': populations,
})
df.sort_values(by='Population', inplace=True)

x_coords = np.arange(len(df))
colors = ['#0000FF' for _ in range(len(df))]
colors[-2] = '#FF0000'
plt.bar(x_coords, df['Population'], tick_label=df['Country'], color=colors)
plt.xticks(rotation=90)
plt.ylabel('Population (Millions)')
plt.title('South American Populations')
plt.show()
```

✓ 0.2s



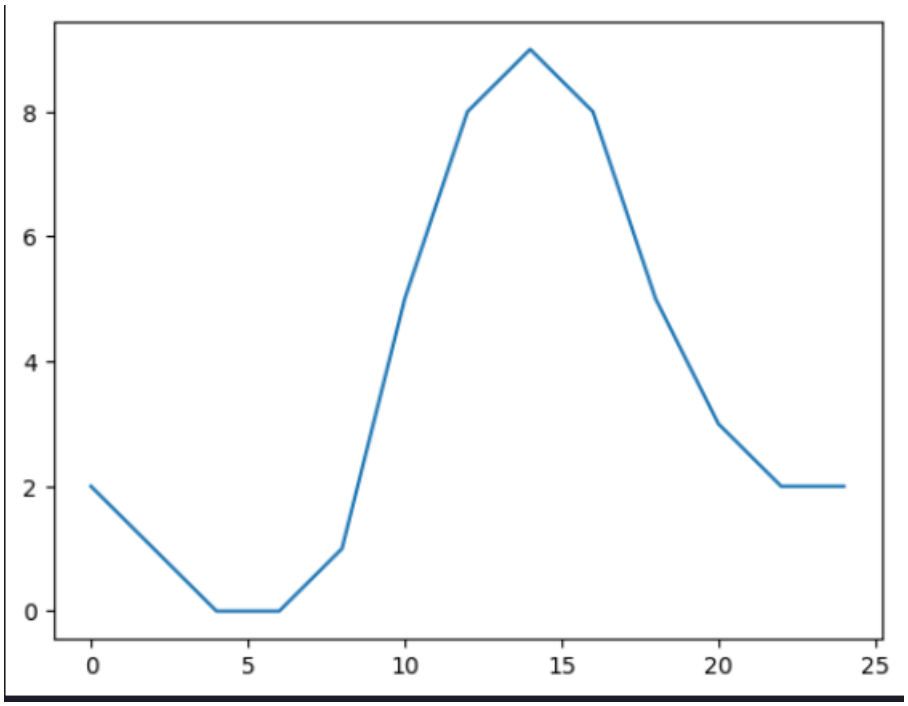
Membuat table menjadi berukurab lebih besar

```
# mengatur ukuran gambar
countries = ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador'
             'Falkland Islands', 'French Guiana', 'Guyana', 'Paraguay', 'Peru
             'Suriname', 'Uruguay', 'Venezuela')

populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,
               3481, 287750, 785409, 7107305, 32880332, 585169, 3470475,
               28258770)

df = pd.DataFrame({
    'Country': countries,
    'Population': populations,
})
df.sort_values(by='Population', inplace=True)

x_coords = np.arange(len(df))
colors = ['#0000FF' for _ in range(len(df))]
colors[-2] = '#FF0000'
plt.figure(figsize=(20,10))
plt.bar(x_coords, df['Population'], tick_label=df['Country'], color=colors)
plt.xticks(rotation=90)
plt.ylabel('Population (Millions)')
plt.title('South American Populations')
plt.show()
```



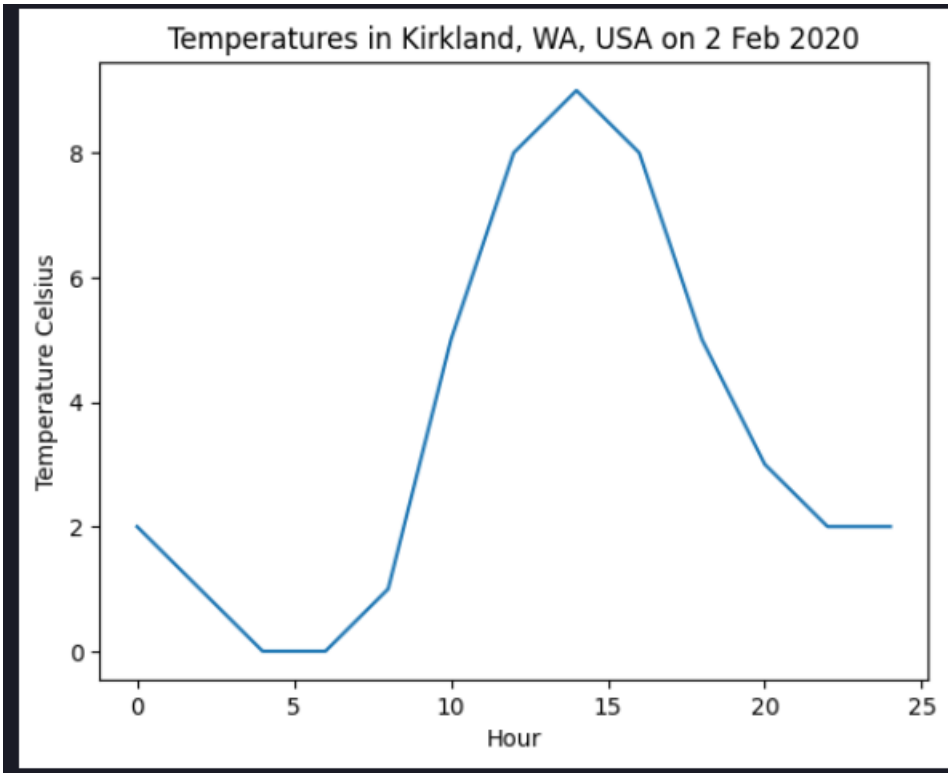
Membuat line graph

```
# memuat data
temperature_c = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

# visualisasi line graph
plt.plot(
  hour,
  temperature_c
)
plt.show()
```



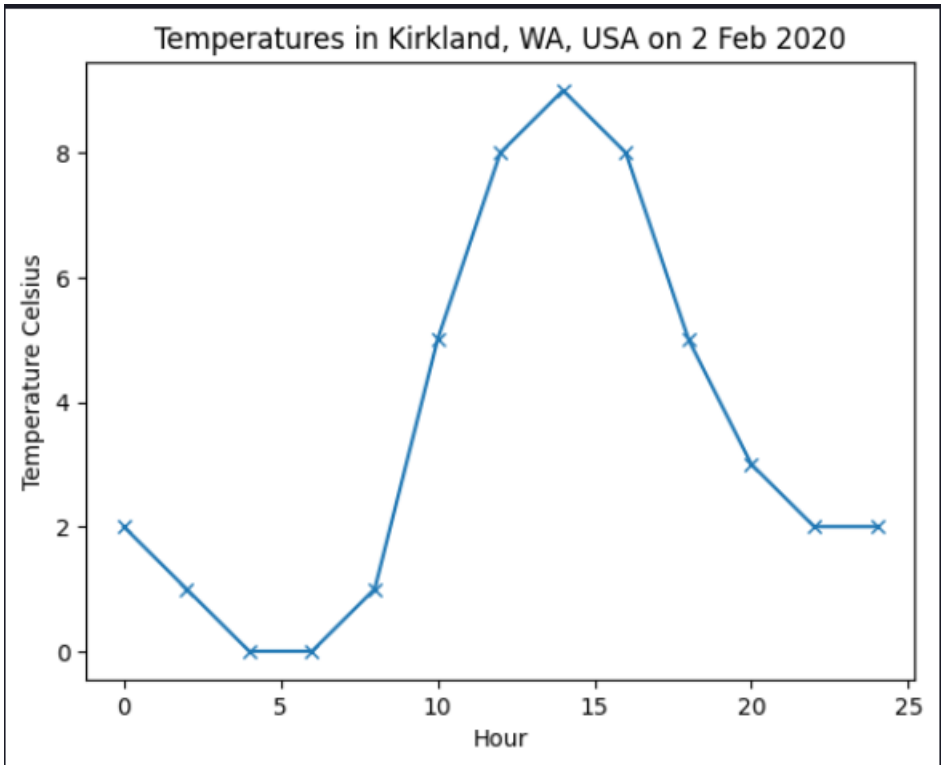
Memberi label dan judul

```
# menambah title(), ylabel(), dan xlabel()
temperature_c = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

plt.plot(
  hour,
  temperature_c,
)
plt.title('Temperatures in Kirkland, WA, USA on 2 Feb 2020')
plt.ylabel('Temperature Celsius')
plt.xlabel('Hour')
plt.show()
```



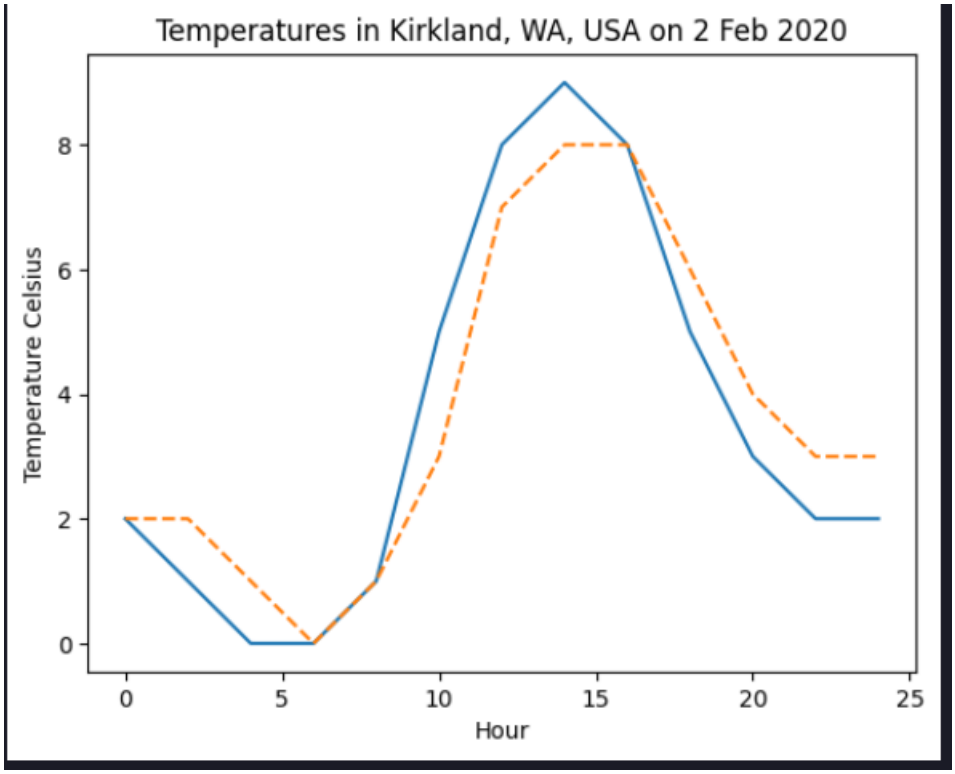
Memberi marker setiap data

```
# menambahkan penanda titik
temperature_c = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

plt.plot(
  hour,
  temperature_c,
  marker='x',
)
plt.title('Temperatures in Kirkland, WA, USA on 2 Feb 2020')
plt.ylabel('Temperature Celsius')
plt.xlabel('Hour')
plt.show()
```



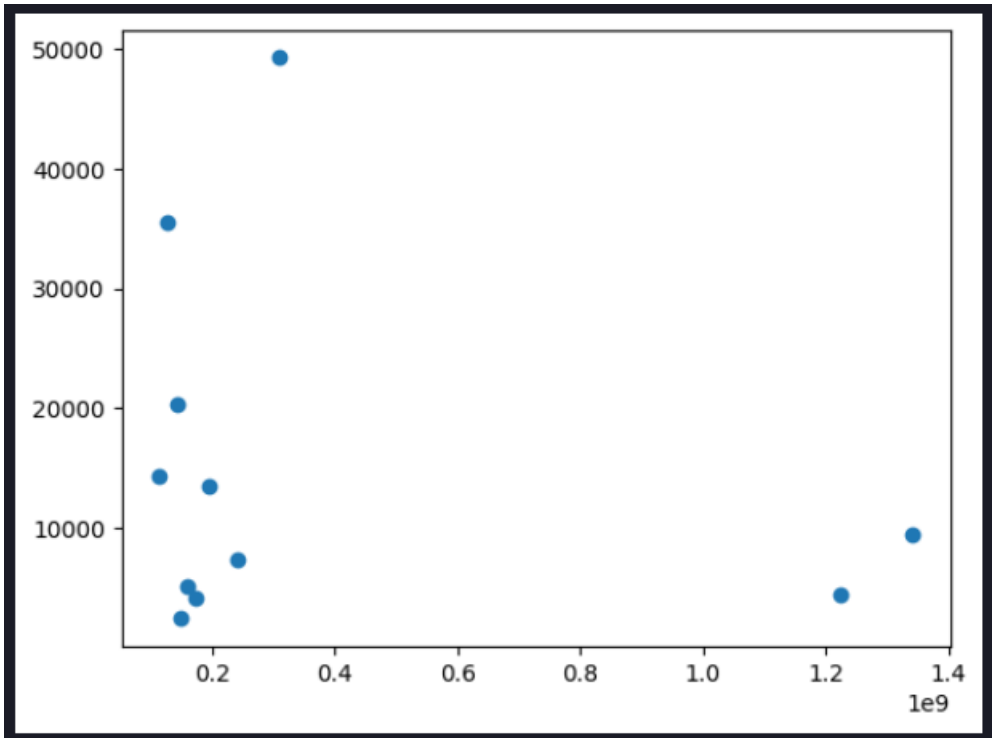
Memberikan line style

```
# visualisasi line graph dengan 2 nilai
temperature_c_actual = [2, 1, 0, 0, 1, 5, 8, 9, 8, 5, 3, 2, 2]
temperature_c_predicted = [2, 2, 1, 0, 1, 3, 7, 8, 8, 6, 4, 3, 3]
hour = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

plt.plot(hour, temperature_c_actual)
plt.plot(hour, temperature_c_predicted, linestyle='--')
plt.title('Temperatures in Kirkland, WA, USA on 2 Feb 2020')
plt.ylabel('Temperature Celsius')
plt.xlabel('Hour')
plt.show()
```



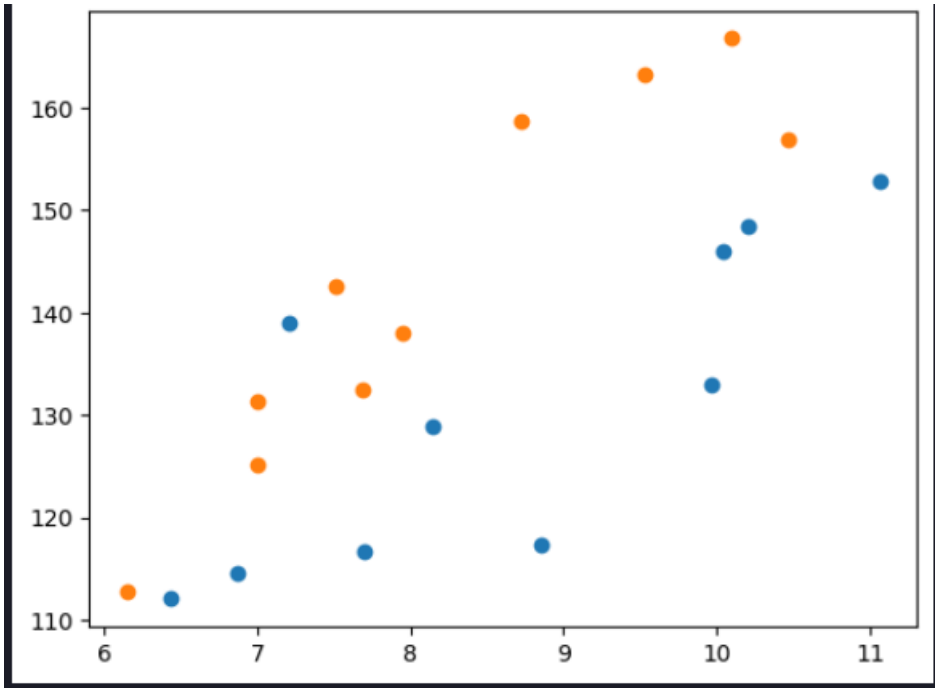
Membuat scatterplot

```
# visualisasi scatter plot
country = ['Bangladesh', 'Brazil', 'China', 'India', 'Indonesia', 'Japan',
           'Mexico', 'Nigeria', 'Pakistan', 'Russia', 'United States']
gdp = [2421, 13418, 9475, 4353, 7378, 35477, 14276, 5087, 4133, 20255, 49267]
population = [148692131, 194946470, 1341335152, 1224614327, 239870937,
              126535920, 113423047, 158423182, 173593383, 142958164, 310383948]

plt.scatter(population, gdp)
plt.show()
```



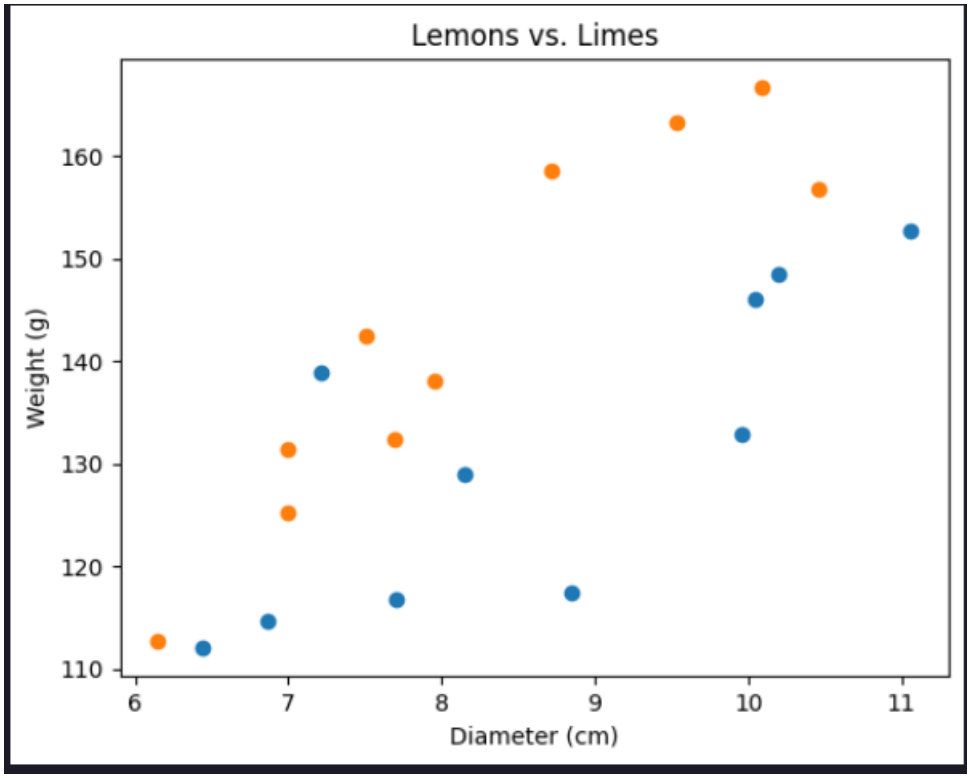
Melakukan plot pada 2 benda

```
# visualisasi scatterplot dengan 2 variabel
lemon_diameter = [6.44, 6.87, 7.7, 8.85, 8.15, 9.96, 7.21, 10.04, 10.2, 11.06]
lemon_weight = [112.05, 114.58, 116.71, 117.4, 128.93,
                132.93, 138.92, 145.98, 148.44, 152.81]

lime_diameter = [6.15, 7.0, 7.0, 7.69, 7.95, 7.51, 10.46, 8.72, 9.53, 10.09]
lime_weight = [112.76, 125.16, 131.36, 132.41, 138.08,
               142.55, 156.86, 158.67, 163.28, 166.74]

plt.scatter(lemon_diameter, lemon_weight)
plt.scatter(lime_diameter, lime_weight)
plt.show()
```



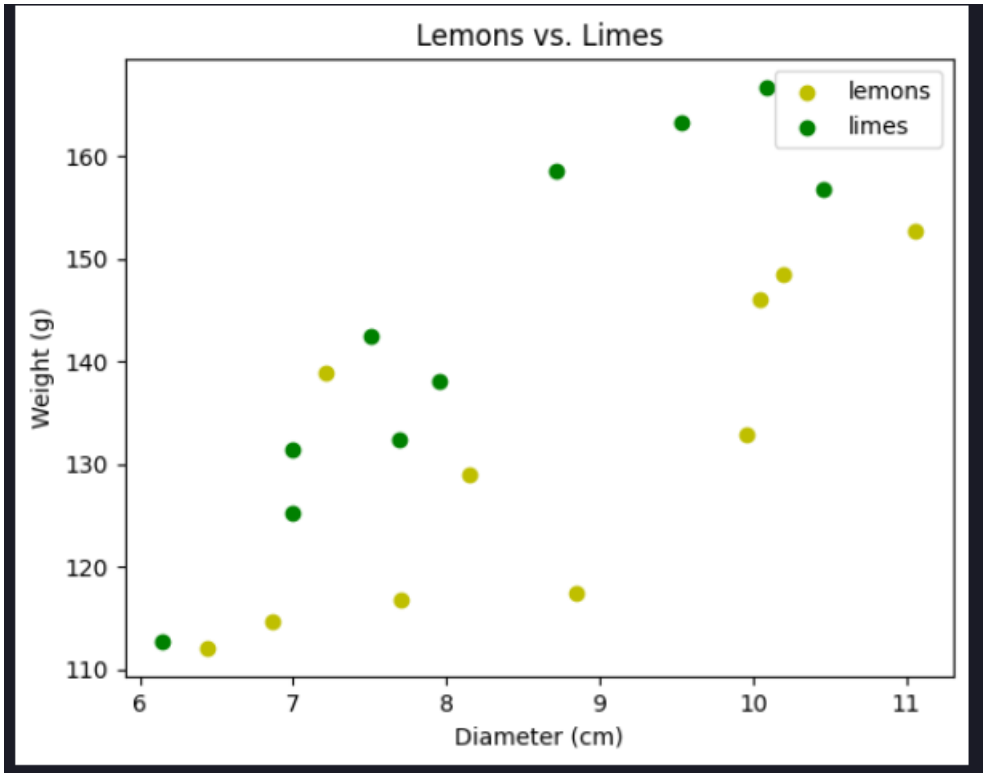
Memberikan label dan judul

```
# menambahkan title, xlabel, dan ylabel
lemon_diameter = [6.44, 6.87, 7.7, 8.85, 8.15, 9.96, 7.21, 10.04, 10.2, 11.06]
lemon_weight = [112.05, 114.58, 116.71, 117.4, 128.93,
                132.93, 138.92, 145.98, 148.44, 152.81]

lime_diameter = [6.15, 7.0, 7.0, 7.69, 7.95, 7.51, 10.46, 8.72, 9.53, 10.09]
lime_weight = [112.76, 125.16, 131.36, 132.41, 138.08,
               142.55, 156.86, 158.67, 163.28, 166.74]

plt.title('Lemons vs. Limes')
plt.xlabel('Diameter (cm)')
plt.ylabel('Weight (g)')
plt.scatter(lemon_diameter, lemon_weight)
plt.scatter(lime_diameter, lime_weight)
plt.show()
```



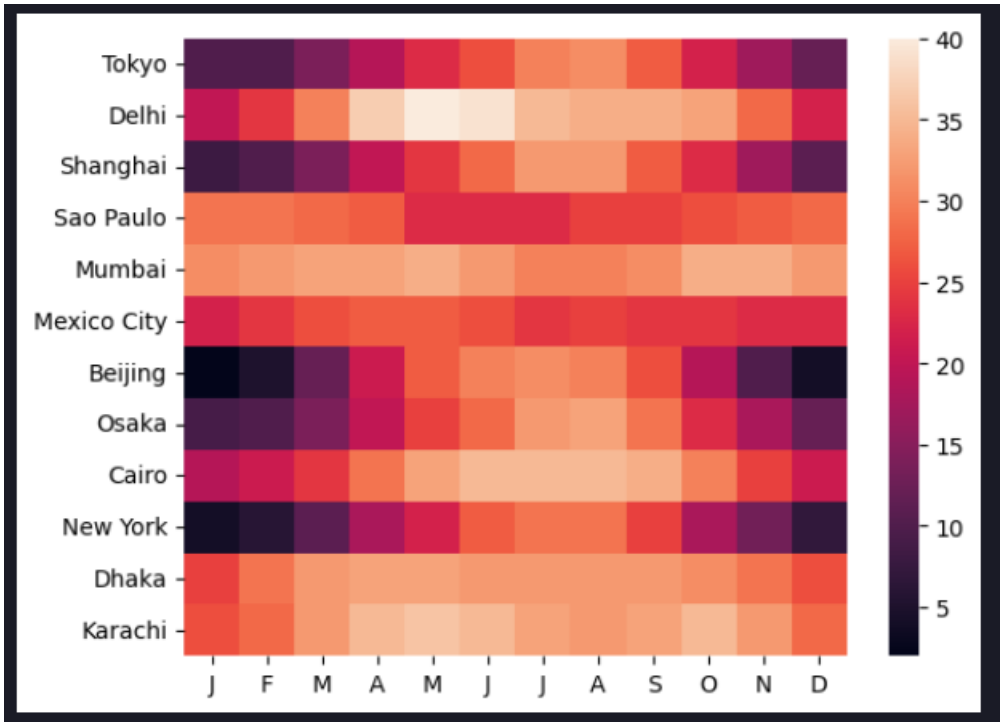
Memberikan warna pada scatterplot

```
# kostumisasi warna scatter plot
lemon_diameter = [6.44, 6.87, 7.7, 8.85, 8.15, 9.96, 7.21, 10.04, 10.2, 11.06]
lemon_weight = [112.05, 114.58, 116.71, 117.4, 128.93,
                132.93, 138.92, 145.98, 148.44, 152.81]

lime_diameter = [6.15, 7.0, 7.0, 7.69, 7.95, 7.51, 10.46, 8.72, 9.53, 10.09]
lime_weight = [112.76, 125.16, 131.36, 132.41, 138.08,
               142.55, 156.86, 158.67, 163.28, 166.74]

plt.title('Lemons vs. Limes')
plt.xlabel('Diameter (cm)')
plt.ylabel('Weight (g)')
plt.scatter(lemon_diameter, lemon_weight, color='y')
plt.scatter(lime_diameter, lime_weight, color='g')
plt.legend(['lemons', 'limes'])
plt.show()
```



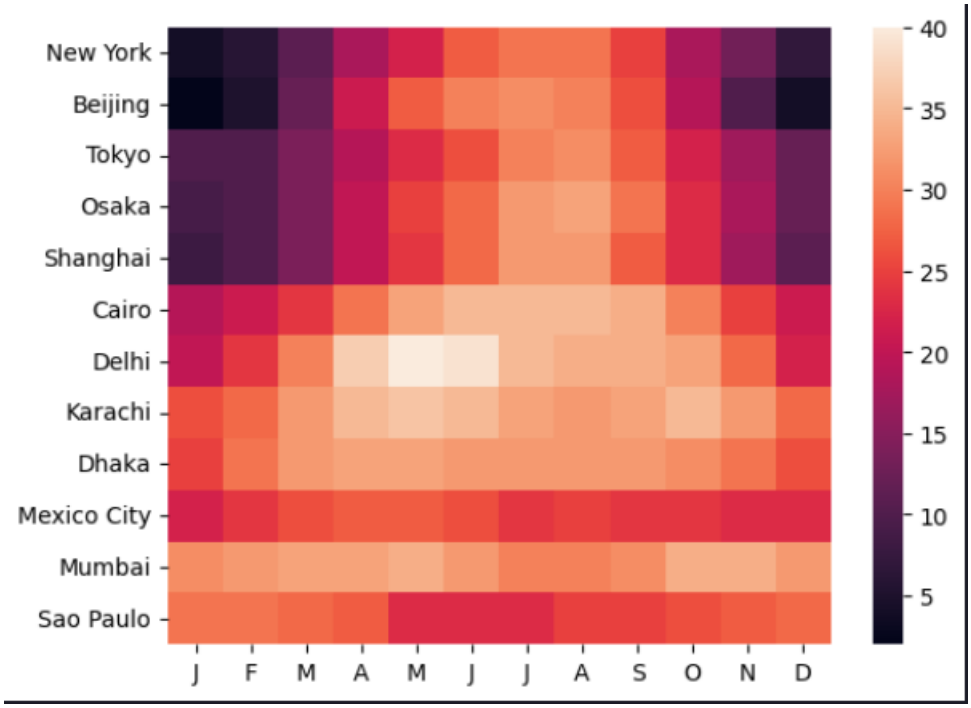
Membuat heatmap

```
# visualisasi heatmap
import seaborn as sns
cities = ['Tokyo', 'Delhi', 'Shanghai', 'Sao Paulo', 'Mumbai', 'Mexico City',
          'Beijing', 'Osaka', 'Cairo', 'New York', 'Dhaka', 'Karachi']
months = ['J', 'F', 'M', 'A', 'M', 'J', 'J', 'A', 'S', 'O', 'N', 'D']
temperatures = [
  [10, 10, 14, 19, 23, 26, 30, 31, 27, 22, 17, 12], # Tokyo
  [20, 24, 30, 37, 40, 39, 35, 34, 34, 33, 28, 22], # Delhi
  [ 8, 10, 14, 20, 24, 28, 32, 32, 27, 23, 17, 11], # Shanghai
  [29, 29, 28, 27, 23, 23, 23, 25, 25, 26, 27, 28], # Sao Paulo
  [31, 32, 33, 33, 34, 32, 30, 30, 31, 34, 34, 32], # Mumbai
  [22, 24, 26, 27, 27, 26, 24, 25, 24, 24, 23, 23], # Mexico City
  [ 2,  5, 12, 21, 27, 30, 31, 30, 26, 19, 10,  4], # Beijing
  [ 9, 10, 14, 20, 25, 28, 32, 33, 29, 23, 18, 12], # Osaka
  [19, 21, 24, 29, 33, 35, 35, 35, 34, 30, 25, 21], # Cairo
  [ 4,  6, 11, 18, 22, 27, 29, 29, 25, 18, 13,  7], # New York
  [25, 29, 32, 33, 33, 32, 32, 32, 32, 31, 29, 26], # Dhaka
  [26, 28, 32, 35, 36, 35, 33, 32, 33, 35, 32, 28], # Karachi
]
sns.heatmap(temperatures, yticklabels=cities, xticklabels=months)
```



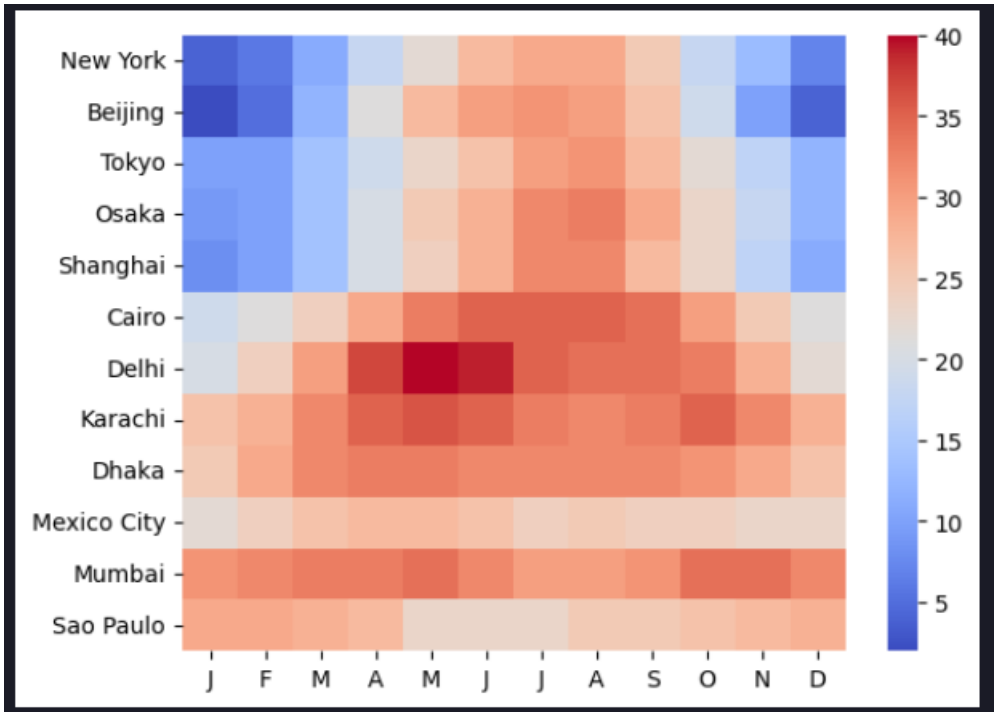
Mengurutkan kota ascending

```
# mengurutkan berdasarkan lintang pada garis bumi
cities = ['New York', 'Beijing', 'Tokyo', 'Osaka', 'Shanghai', 'Cairo', 'Delhi',
          'Karachi', 'Dhaka', 'Mexico City', 'Mumbai', 'Sao Paulo']

temperatures = [
  [ 4,  6, 11, 18, 22, 27, 29, 29, 25, 18, 13,  7], # New York
  [ 2,  5, 12, 21, 27, 30, 31, 30, 26, 19, 10,  4], # Beijing
  [10, 10, 14, 19, 23, 26, 30, 31, 27, 22, 17, 12], # Tokyo
  [ 9, 10, 14, 20, 25, 28, 32, 33, 29, 23, 18, 12], # Osaka
  [ 8, 10, 14, 20, 24, 28, 32, 32, 27, 23, 17, 11], # Shanghai
  [19, 21, 24, 29, 33, 35, 35, 35, 34, 30, 25, 21], # Cairo
  [20, 24, 30, 37, 40, 39, 35, 34, 34, 33, 28, 22], # Delhi
  [26, 28, 32, 35, 36, 35, 33, 32, 33, 35, 32, 28], # Karachi
  [25, 29, 32, 33, 33, 32, 32, 32, 32, 31, 29, 26], # Dhaka
  [22, 24, 26, 27, 27, 26, 24, 25, 24, 24, 23, 23], # Mexico City
  [31, 32, 33, 33, 34, 32, 30, 30, 31, 34, 34, 32], # Mumbai
  [29, 29, 28, 27, 23, 23, 23, 25, 25, 26, 27, 28], # Sao Paulo
]

sns.heatmap(temperatures, yticklabels=cities, xticklabels=months)
#<matplotlib.axes._subplots.AxesSubplot at 0x22345cc0a48>
```



Mengubah warna pada heatmap

```
# mengubah skema warna dengan cmap
cities = ['New York', 'Beijing', 'Tokyo', 'Osaka', 'Shanghai', 'Cairo', 'Delh
          'Karachi', 'Dhaka', 'Mexico City', 'Mumbai', 'Sao Paulo']

temperatures = [
  [ 4,  6, 11, 18, 22, 27, 29, 29, 25, 18, 13,  7], # New York
  [ 2,  5, 12, 21, 27, 30, 31, 30, 26, 19, 10,  4], # Beijing
  [10, 10, 14, 19, 23, 26, 30, 31, 27, 22, 17, 12], # Tokyo
  [ 9, 10, 14, 20, 25, 28, 32, 33, 29, 23, 18, 12], # Osaka
  [ 8, 10, 14, 20, 24, 28, 32, 32, 27, 23, 17, 11], # Shanghai
  [19, 21, 24, 29, 33, 35, 35, 35, 34, 30, 25, 21], # Cairo
  [20, 24, 30, 37, 40, 39, 35, 34, 34, 33, 28, 22], # Delhi
  [26, 28, 32, 35, 36, 35, 33, 32, 33, 35, 32, 28], # Karachi
  [25, 29, 32, 33, 33, 32, 32, 32, 32, 31, 29, 26], # Dhaka
  [22, 24, 26, 27, 27, 26, 24, 25, 24, 24, 23, 23], # Mexico City
  [31, 32, 33, 33, 34, 32, 30, 30, 31, 34, 34, 32], # Mumbai
  [29, 29, 28, 27, 23, 23, 23, 25, 25, 26, 27, 28], # Sao Paulo
]

sns.heatmap(
    temperatures,
    yticklabels=cities,
    xticklabels=months,
    cmap='coolwarm',
)
```



Memuat data

```
# memuat data
path='https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/automobileEDA.csv'
df = pd.read_csv(path)
df.head()
```

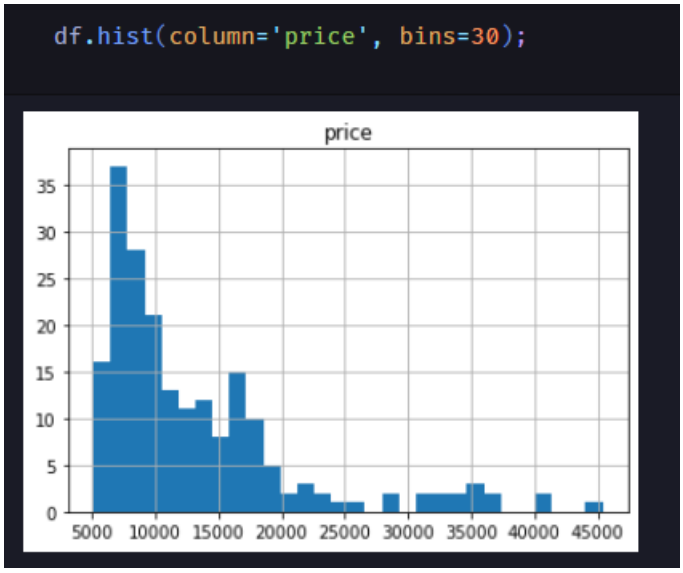
Python

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-L/100km	horsepower-binned	diesel	gas
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27	13495.0	11.190476	Medium	0	1
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27	16500.0	11.190476	Medium	0	1
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...	9.0	154.0	5000.0	19	26	16500.0	12.368421	Medium	0	1
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	...	10.0	102.0	5500.0	24	30	13950.0	9.791667	Medium	0	1
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	...	8.0	115.0	5500.0	18	22	17450.0	13.055556	Medium	0	1

5 rows × 29 columns

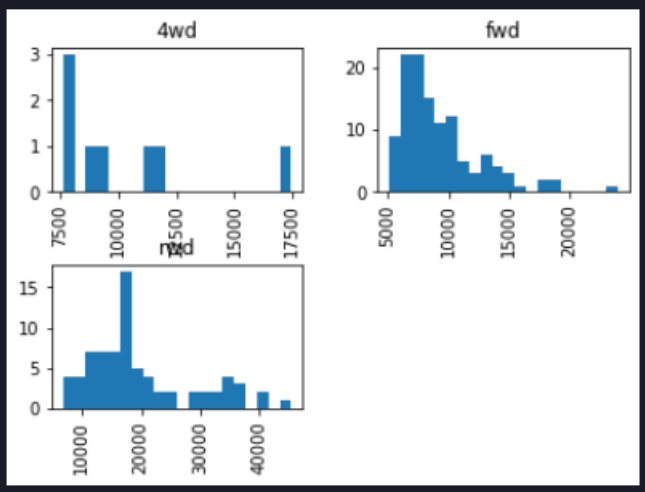
Visualisasi histogram

```
df.hist(column='price', bins=30);
```



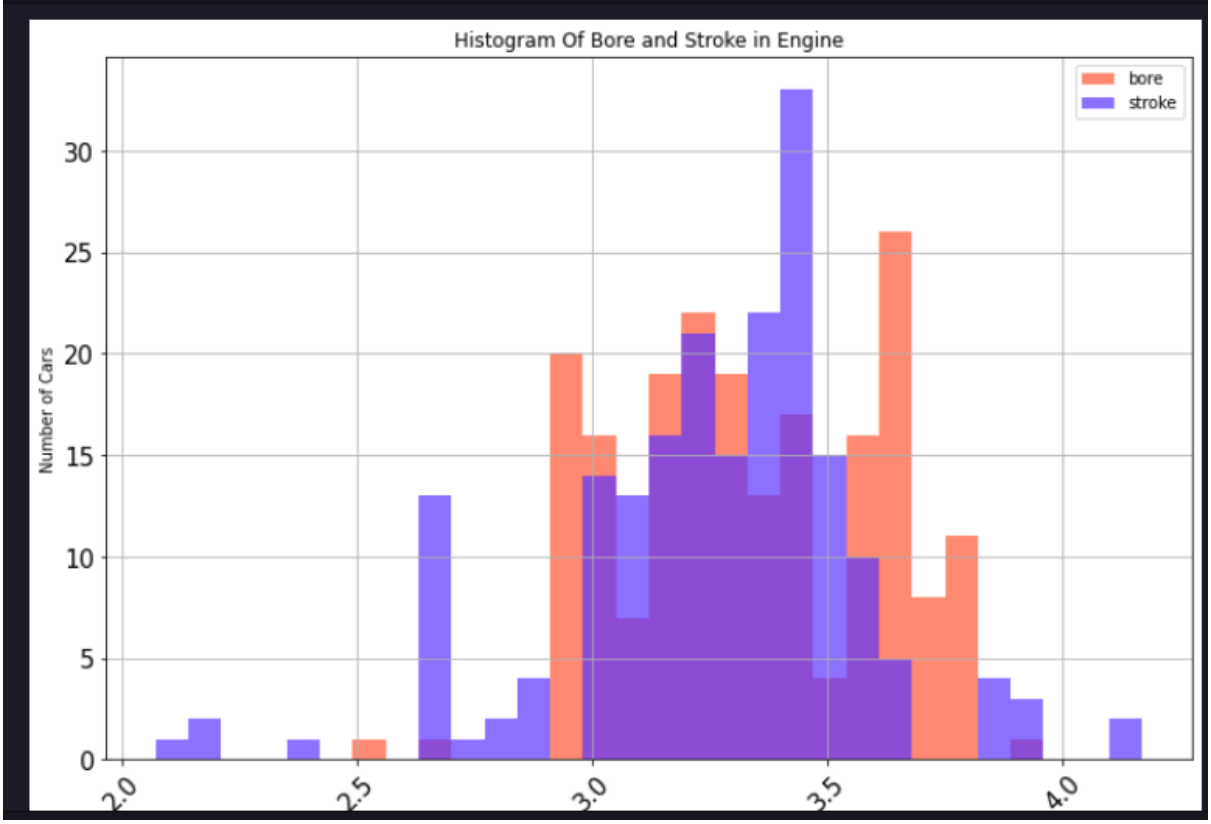
Melakukan plot grup

```
# melakukan plot beberapa grup
df.hist(column='price', by='drive-wheels', bins=20);
```



Melakukan plot pada series

```
# melakukan plot pada beberapa seri
df[['bore','stroke']].plot(kind='hist',
        alpha=0.7,
        bins=30,
        title='Histogram Of Bore and Stroke in Engine',
        rot=45,
        grid=True,
        figsize=(12,8),
        fontsize=15,
        color=['#FF5733', '#5C33FF'])
plt.xlabel('Size')
plt.ylabel("Number of Cars");
```



Import library scipy

```
# import library scipy
from scipy import stats
```

Menghitung koefisi korelasi

```
# menghitung koefisien korelasi pearson dari 'wheels-base' & 'price'
pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.5846418222655083  with a P-value of P = 8.076488270732873e-20
```

```python
# menghitung koefisien korelasi pearson dari 'horsepower' & 'price'
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P = ", p_value)
```

```
The Pearson Correlation Coefficient is 0.8095745670036555  with a P-value of P =  6.369057428261186e-48
```

Daftar type data dataframe

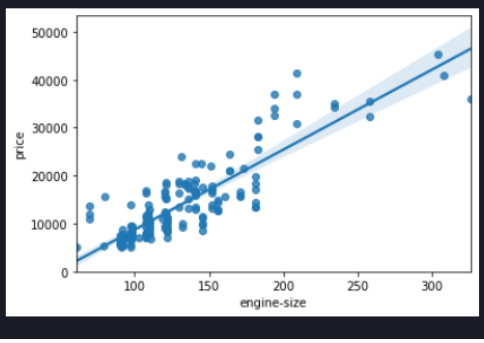
```python
# daftar tipe data setiap kolom
print(df.dtypes)
```

```
symboling              int64
normalized-losses      int64
make                  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                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
 ...
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

Visualisasi scatterplot dengan regplot

```
# visualisasi scatter plot menggunakan regplot untuk mengetahui hubungan korelasi variabel 'engine-size' & 'price'
sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

```
(0.0, 53316.987813911066)
```



Menentukan korelasi column

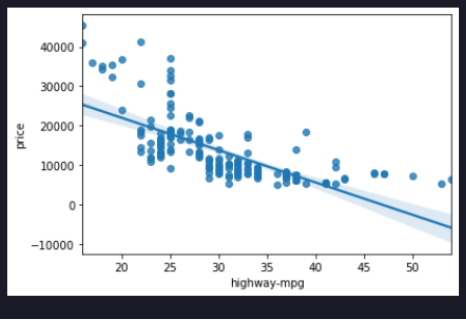
```
# mengetahui korelasi 'engine-size' & 'price'
df[["engine-size", "price"]].corr()
```

	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

Menggunakan regplot untuk mengetahui korelasi

```
# visualisasi scatter plot menggunakan regplot untuk mengetahui hubungan korelasi variabel 'engine-size' & 'price'
sns.regplot(x="highway-mpg", y="price", data=df)
```

```
<AxesSubplot:xlabel='highway-mpg', ylabel='price'>
```



Menentukan hubungan korelasi

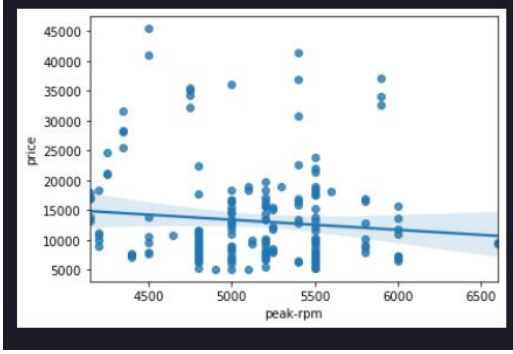
```
# mengetahui korelasi 'highway-mpg' & 'price'
df[['highway-mpg', 'price']].corr()
```

	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

Visualisasi scatterplot

```
# visualisasi scatter plot menggunakan regplot untuk mengetahui hubungan korelasi variabel 'peak-rpm' & 'price'
sns.regplot(x="peak-rpm", y="price", data=df)
```

```
<AxesSubplot:xlabel='peak-rpm', ylabel='price'>
```



Memeriksa korelasi

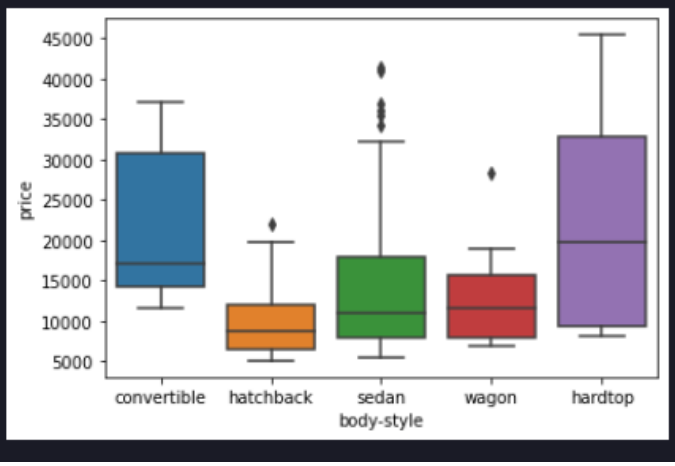
```
df[['peak-rpm','price']].corr()
```

	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

Penggunaan boxplot Visualisasi variable

```
# visualisasi variabel kategori statistik variabel 'body-style' & 'price'
sns.boxplot(x="body-style", y="price", data=df)
```

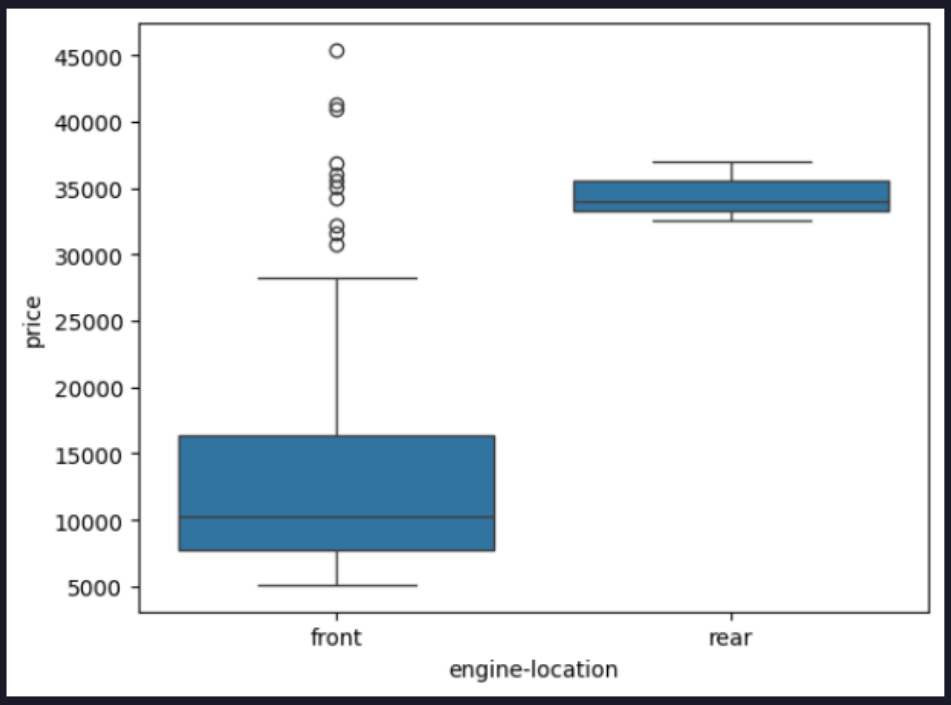
```
<AxesSubplot:xlabel='body-style', ylabel='price'>
```




```
# visualisasi variabel kategori statistik variabel 'engine-location' & 'price'
sns.boxplot(x="engine-location", y="price", data=df)
```

✓ 0.1s

```
<Axes: xlabel='engine-location', ylabel='price'>
```



Deskripsi pada data

```python
# melakukan describe data
df.describe()
```

0.0s Python

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	L/
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	197.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.
mean	0.840796	122.00000	98.797015	0.837102	0.915126	53.766667	2555.666667	126.875622	3.330692	3.256904	10.164279	103.405534	5117.665368	25.179104	30.686567	13207.129353	9.
std	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.296727	41.546834	0.268072	0.319256	4.004965	37.365700	478.113805	6.423220	6.815150	7947.066342	2.
min	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.000000	5118.000000	4.
25%	0.000000	101.00000	94.500000	0.801538	0.890278	52.000000	2169.000000	98.000000	3.150000	3.110000	8.600000	70.000000	4800.000000	19.000000	25.000000	7775.000000	7.
50%	1.000000	122.00000	97.000000	0.832292	0.909722	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000	5125.369458	24.000000	30.000000	10295.000000	9.
75%	2.000000	137.00000	102.400000	0.881788	0.925000	55.500000	2926.000000	141.000000	3.580000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.000000	16500.000000	12.
max	3.000000	256.00000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	262.000000	6600.000000	49.000000	54.000000	45400.000000	18.

Pengaturan default "describe" melewatkan variabel tipe objek. Kita bisa menggunakan code ini untuk menghitung jumlah type data objek

```python
# melakukan describe data terhadap variabel tipe objek
df.describe(include=['object'])
```

0.0s Python

	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system	horsepower-binned
count	201	201	201	201	201	201	201	201	201	200
unique	22	2	2	5	3	2	6	7	8	3
top	toyota	std	four	sedan	fwd	front	ohc	four	mpfi	Low
freq	32	165	115	94	118	198	145	157	92	115

Menghitung unit variable

Nilai-hitungan adalah cara untuk memahami berapa banyak unit dari setiap karakteristik/variabel yang kita miliki. Kita bisa menerapkan metode `"value_counts"` pada kolom 'drive-wheels'. Jangan lupa metode `"value_counts"` hanya berfungsi pada seri Pandas, bukan Pandas Dataframes.

```python
# menghitung nilai pada 'drive-wheels'
df['drive-wheels'].value_counts()
```

0.0s Python

```
drive-wheels
fwd    118
rwd     75
4wd      8
Name: count, dtype: int64
```

Kita dapat mengonversi data series ke Dataframe sebagai berikut:

```python
# konversi series ke dataframe
df['drive-wheels'].value_counts().to_frame()
```

0.0s Python

drive-wheels	count
fwd	118
rwd	75
4wd	8

Mengubah nama variable

```python
# membuat dataframe baru dengan nama "drive_wheels_counts"
drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()

# mengganti nama "drive-wheels" menjadi "value_counts"
drive_wheels_counts.rename(columns={'drive-wheels': 'value_counts'}, inplace=True)
drive_wheels_counts
```

✓ 0.0s

drive-wheels	count
fwd	118
rwd	75
4wd	8

Sekarang mari kita ganti nama indeks menjadi 'drive-wheels':

```python
# mengganti nama indeks menjadi 'drive-wheels'
drive_wheels_counts.index.name = 'drive-wheels'
drive_wheels_counts
```

✓ 0.0s

drive-wheels	count
fwd	118
rwd	75
4wd	8

Kita dapat mengulangi proses di atas untuk variabel 'engine-location'.

```python
# mengulang proses di atas dengan dataframe baru 'engine-location'
engine_loc_counts = df['engine-location'].value_counts().to_frame()
engine_loc_counts.rename(columns={'engine-location': 'value_counts'}, inplace=True)
engine_loc_counts.index.name = 'engine-location'
engine_loc_counts.head(10)
```

✓ 0.0s

engine-location	count
front	198
rear	3

Mencari value unique

```python
# menampilkan nilai unik data series
df['drive-wheels'].unique()
```

✓ 0.0s

```
array(['rwd', 'fwd', '4wd'], dtype=object)
```

Membuat column baru

```
# membuat dataframe baru dengna kolom 'drive-wheels','body-style', dan 'price'
df_group_one = df[['drive-wheels','body-style','price']]
```

0.0s

Melakukan grouping rata2

```
# melakukan grouping
df_group_one = df_group_one.groupby(['drive-wheels'],as_index=False)['price'].mean()
df_group_one
```

0.0s

	drive-wheels	price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

Memberi nilai default 0 untuk null value

```
# melakukan pivot data dengan handle missing value dengan nilai 0
grouped_pivot = grouped_pivot.fillna(0) #fill missing values with 0
grouped_pivot
```

0.0s

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

Melakukan groping body style terhadap harga

```
# melakukan grouping dari  "body-style" berdasarkan rata-rata "harga
df_gptest2 = df[['body-style','price']]
grouped_test_bodystyle = df_gptest2.groupby(['body-style'],as_index= False).mean()
grouped_test_bodystyle
```

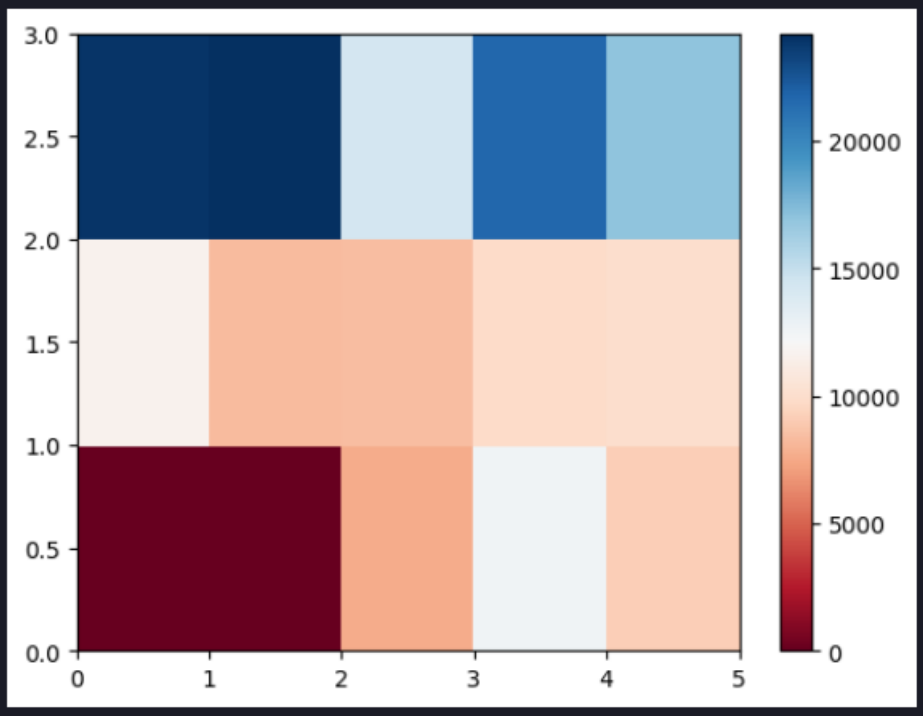
0.0s

	body-style	price
0	convertible	21890.500000
1	hardtop	22208.500000
2	hatchback	9957.441176
3	sedan	14459.755319
4	wagon	12371.960000

Visualisasi heatmap

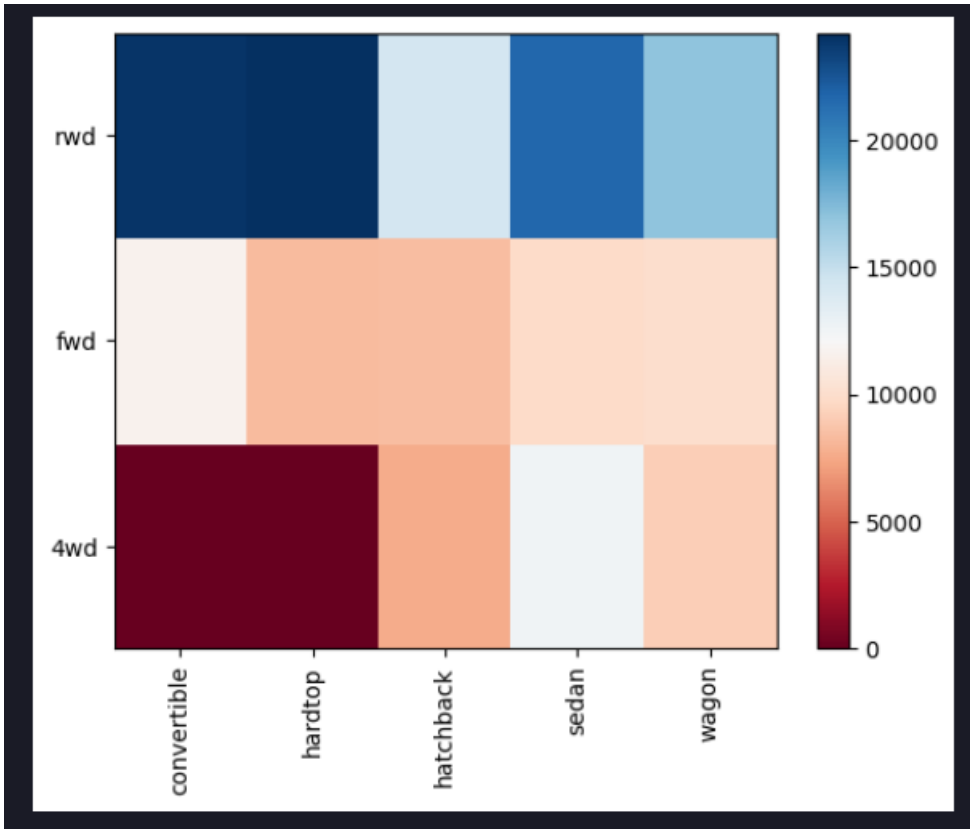
```
#visualisasi heatmap
plt.pcolor(grouped_pivot, cmap='RdBu')
plt.colorbar()
plt.show()
```

✓ 0.1s



Menampilkan heatmap

```
fig, ax = plt.subplots()
im = ax.pcolor(grouped_pivot, cmap='RdBu')

# nama label
row_labels = grouped_pivot.columns.levels[1]
col_labels = grouped_pivot.index

# pindahkan ticks and labels ke tenah
ax.set_xticks(np.arange(grouped_pivot.shape[1]) + 0.5, minor=False)
ax.set_yticks(np.arange(grouped_pivot.shape[0]) + 0.5, minor=False)

# masukan  labels
ax.set_xticklabels(row_labels, minor=False)
ax.set_yticklabels(col_labels, minor=False)

# rotasi label
plt.xticks(rotation=90)

fig.colorbar(im)
plt.show()
```



Melakukan gruping

```
# mengelompokan data 'drive-wheels' & 'price' berdasarkan 'drive-wheels'
grouped_test2=df_gptest[['drive-wheels', 'price']].groupby(['drive-wheels'])
grouped_test2.head(2)
```

✓ 0.0s

	drive-wheels	price
0	rwd	13495.0
1	rwd	16500.0
3	fwd	13950.0
4	4wd	17450.0
5	fwd	15250.0
136	4wd	7603.0

Menunjukan data

```
# tampilkan data
df_gptest
```

✓ 0.0s

	drive-wheels	body-style	price
0	rwd	convertible	13495.0
1	rwd	convertible	16500.0
2	rwd	hatchback	16500.0
3	fwd	sedan	13950.0
4	4wd	sedan	17450.0
...	...	...	...
196	rwd	sedan	16845.0
197	rwd	sedan	19045.0
198	rwd	sedan	21485.0
199	rwd	sedan	22470.0
200	rwd	sedan	22625.0

201 rows × 3 columns

Mendapatkan f score dan p score

kita dapat menggunakan fungsi `'f_oneway'` di modul 'stats' untuk mendapatkan F-Score dan P-Value

```
# ANOVA
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'], grouped_test2.get_group('rwd')['price'], grouped_test2.get_group('4wd')['price'])

print( "ANOVA results: F=", f_val, ", P =", p_val)
```

✓ 0.0s

```
ANOVA results: F= 67.95406500780399 , P = 3.3945443577149576e-23
```

Hasil ANOVA ini termasuk hasil yang bagus, dengan F-Score yang besar menunjukkan korelasi yang kuat dan nilai P hampir 0 menyiratkan signifikansi statistik yang hampir pasti. Tetapi apakah ini berarti ketiga kelompok yang diuji semuanya berkorelasi tinggi?

Separately: fwd and rwd

```
# menampilkan f-score dan p-value'
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'], grouped_test2.get_group('rwd')['price'])

print( "ANOVA results: F=", f_val, ", P =", p_val )
```

✓ 0.0s Python

```
ANOVA results: F= 130.5533160959111 , P = 2.2355306355677366e-23
```

4wd and fwd

```
# menampilkan f-score dan p-value
f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'], grouped_test2.get_group('fwd')['price'])

print("ANOVA results: F=", f_val, ", P =", p_val)
```

✓ 0.0s

```
ANOVA results: F= 0.665465750252303 , P = 0.4162011669784502
```