Introduction to Pandas

Memanggil Library Pandas

Pandas adalah library python open source yang biasanya digunakan untuk kebutuhan data analisis. Pandas membuat Python supaya dapat bekerja dengan data yang berbentuk tabular sepert spreadsheet dengan cara pemuatan data yang tepat, manipulasi data, menggabungkan data, serta ada berbagai fungsi yang lain

In [ ]:

```
import pandas as pd
import numpy as np
```

DataFrame & Series

In [ ]:

```
import pandas as pd
# Series
number_list = pd.Series([1,2,3,4,5,6])
print("Series:")
print(number_list)

# DataFrame
matrix = [[1,2,3],
          ['a','b','c'],
          [3,4,5],
          ['d',4,6]]
matrix_list = pd.DataFrame(matrix)
print("DataFrame:")
print(matrix_list)
```

```
Series:
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
DataFrame:
   0  1  2
0  1  2  3
1  a  b  c
2  3  4  5
3  d  4  6
```

Atribut DataFrame & Series - Part 1

In [ ]:

```
import pandas as pd

# Series
number_list = pd.Series([1,2,3,4,5,6])

# DataFrame
matrix_list = pd.DataFrame([[1,2,3],
                            ['a','b','c'],
                            [3,4,5],
                            ['d',4,6]])

# [1] attribute .info()
print("[1] attribute .info()")
```

```
print(matrix_list.info())

# [2] attribute .shape
print("[2] attribute .shape")
print("Shape dari number_list:", number_list.shape)
print("Shape dari matrix_list:", matrix_list.shape)

# [3] attribute .dtypes
print("attribute .dtypes")
print("Tipe data dari number_list:", number_list.dtypes)
print("Tipe data dari matrix_list:", matrix_list.dtypes)

# [4] attribute .astype()
print("attribute .astype()")
print("Konversi number_list ke str:", number_list.astype("str"))
print("Konversi matrix_list ke str:", matrix_list.astype("str"))
```

```
[1] attribute .info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       4 non-null      object
 1   1       4 non-null      object
 2   2       4 non-null      object
dtypes: object(3)
memory usage: 224.0+ bytes
None
[2] attribute .shape
Shape dari number_list: (6,)
Shape dari matrix_list: (4, 3)
attribute .dtypes
Tipe data dari number_list: int64
Tipe data dari matrix_list: 0    object
1    object
2    object
dtype: object
attribute .astype()
Konversi number_list ke str: 0    1
1    2
2    3
3    4
4    5
5    6
dtype: object
Konversi matrix_list ke str:    0  1  2
0  1  2  3
1  a  b  c
2  3  4  5
3  d  4  6
```

Atribut DataFrame & Series - Part 2

In [ ]:

```
import pandas as pd

# Series
number_list = pd.Series([1,2,3,4,5,6])

# DataFrame
matrix_list = pd.DataFrame([[1,2,3],
                            ['a','b','c'],
                            [3,4,5],
                            ['d',4,6]])

# [5] attribute .copy()
print("[5] attribute .copy()")
num_list = number_list.copy()
print("Copy number_list ke num_list:", num_list)
```

```
mtr_list = matrix_list.copy()
print("Copy number_list ke num_list:", mtr_list)

# [6] attribute .to_list()
print("[6] attribute .to_list()")
print(number_list.to_list())

# [7] attribute .unique()
print("[7] attribute .unique()")
print(number_list.unique())
```

```
[5] attribute .copy()
Copy number_list ke num_list: 0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
Copy number_list ke num_list:    0  1  2
0  1  2  3
1  a  b  c
2  3  4  5
3  d  4  6
[6] attribute .to_list()
[1, 2, 3, 4, 5, 6]
[7] attribute .unique()
[1 2 3 4 5 6]
```

Atribut DataFrame & Series - Part 3

In [ ]:

```
import pandas as pd

# Series
number_list = pd.Series([1,2,3,4,5,6])

# DataFrame
matrix_list = pd.DataFrame([[1,2,3],
                            ['a','b','c'],
                            [3,4,5],
                            ['d',4,6]])

# [8] attribute .index
print("[8] attribute .index")
print("Index number_list:", number_list.index)
print("Index matrix_list:", matrix_list.index)

# [9] attribute .columns
print("[9] attribute .columns")
print("Columns matrix_list:", matrix_list.columns)

# [10] attribute .loc
print("[10] attribute .loc")
print(".loc[0:1] pada number_list:", number_list.loc[0:1])
print(".loc[0:1] pada matrix_list:", matrix_list.loc[0:1])

# [11] attribute .iloc
print("[11] attribute .iloc")
print(".iloc[0:1] pada number_list:", number_list.iloc[0:1])
print(".iloc[0:1] pada matrix_list:", matrix_list.iloc[0:1])
```

```
[8] attribute .index
Index number_list: RangeIndex(start=0, stop=6, step=1)
Index matrix_list: RangeIndex(start=0, stop=4, step=1)
[9] attribute .columns
Columns matrix_list: RangeIndex(start=0, stop=3, step=1)
[10] attribute .loc
.loc[0:1] pada number_list: 0    1
1    2
dtype: int64
```

```
dtype: int64
.loc[0:1] pada matrix_list:    0  1  2
0  1  2  3
1  a  b  c
[11] attribute .iloc
.iloc[0:1] pada number_list: 0    1
dtype: int64
.iloc[0:1] pada matrix_list:    0  1  2
0  1  2  3
```

Quiz

In [ ]:

```
matrix = [[1,2,3],
          ['a','b','c'],
          [3,4,5],
          ['d',4,6]]
matrix_list = pd.DataFrame(matrix)

matrix_list.iloc[0:2,2].to_list()
```

Out[ ]:

```
[3, 'c']
```

Creating Series & DataFrame from List

In [ ]:

```
import pandas as pd

# Creating series from list
ex_list = ['a',1,3,5,'c','d']
ex_series = pd.Series(ex_list)
print(ex_series)

# Creating dataframe from list of list
ex_list_of_list = [[1  ,'a','b' ,'c'],
                   [2.5,'d','e' ,'f'],
             [5  ,'g','h' ,'i'],
             [7.5,'j',10.5,'l']]
index = ['dq','lab','kar','lan']
cols = ['float','char','obj','char']
ex_df = pd.DataFrame(ex_list_of_list, index=index, columns=cols)
print(ex_df)
```

```
0    a
1    1
2    3
3    5
4    c
5    d
dtype: object
     float char   obj char
dq     1.0    a     b    c
lab    2.5    d     e    f
kar    5.0    g     h    i
lan    7.5    j  10.5    l
```

Creating Series & DataFrame from Dictionary

In [ ]:

```
import pandas as pd

# Creating series from dictionary
dict_series = {'1':'a',
      '2':'b',
      '3':'c'}
```

```
ex_series = pd.Series(dict_series)
print(ex_series)

# Creating dataframe from dictionary
df_series = {'1':['a','b','c'],
             '2':['b','c','d'],
             '4':[2,3,'z']}
ex_df = pd.DataFrame(df_series)
print(ex_df)
```

```
1    a
2    b
3    c
dtype: object
   1  2  4
0  a  b  2
1  b  c  3
2  c  d  z
```

Creating Series & DataFrame fron Numpy Array

In [ ]:

```
import pandas as pd
import numpy as np

# Creating series from numpy array (1D)
arr_series = np.array([1,2,3,4,5,6,6,7])
ex_series = pd.Series(arr_series)
print(ex_series)

# Creating dataframe from numpy array (2D)
arr_df = np.array([[1  ,2  ,3  ,5],
                   [5  ,6  ,7  ,8],
                   ['a','b','c',10]])
ex_df = pd.DataFrame(arr_df)
print(ex_df)
```

```
0    1
1    2
2    3
3    4
4    5
5    6
6    6
7    7
dtype: int64
   0  1  2   3
0  1  2  3   5
1  5  6  7   8
2  a  b  c  10
```

Dataset I/O

Read Dataset - CSV dan TSV

In [ ]:

```
import pandas as pd

# File CSV
df_csv = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")
print(df_csv.head(3)) # Menampilkan 3 data teratas

# File TSV
df_tsv = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_tsv.tsv", sep='\t')
print(df_tsv.head(3)) # Menampilkan 3 data teratas
```

```
   order_id  order_date  customer_id  ...    brand quantity item_price
0   1612339  2019-01-01        18055  ...  BRAND_C        4    1934000
1   1612339  2019-01-01        18055  ...  BRAND_V        8     604000
2   1612339  2019-01-01        18055  ...  BRAND_G       12     747000

[3 rows x 9 columns]
   order_id  order_date  customer_id  ...    brand quantity item_price
0   1612339  2019-01-01        18055  ...  BRAND_C        4    1934000
1   1612339  2019-01-01        18055  ...  BRAND_V        8     604000
2   1612339  2019-01-01        18055  ...  BRAND_G       12     747000

[3 rows x 9 columns]
```

Read Dataset - Excel

In [ ]:

```
import pandas as pd

# File xlsx dengan data di sheet "test"
df_excel = pd.read_excel("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_excel.xlsx", sheet_name="test")
print(df_excel.head(4)) # Menampilkan 4 data teratas
```

```
   order_id  order_date  customer_id  ...    brand quantity item_price
0   1612339  2019-01-01        18055  ...  BRAND_C        4    1934000
1   1612339  2019-01-01        18055  ...  BRAND_V        8     604000
2   1612339  2019-01-01        18055  ...  BRAND_G       12     747000
3   1612339  2019-01-01        18055  ...  BRAND_B       12     450000

[4 rows x 9 columns]
```

Read Dataset - JSON

In [ ]:

```
import pandas as pd

# File JSON
url = "https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/covid2019-api-herokuapp-v2.json"
df_json = pd.read_json(url)
print(df_json.head(10)) # Menampilkan 10 data teratas
```

```
                                                data          dt          ts
0  {'location': 'US', 'confirmed': 3363056, 'deat...  07-14-2020  1594684800
1  {'location': 'Brazil', 'confirmed': 1884967, '...  07-14-2020  1594684800
2  {'location': 'India', 'confirmed': 906752, 'de...  07-14-2020  1594684800
3  {'location': 'Russia', 'confirmed': 732547, 'd...  07-14-2020  1594684800
4  {'location': 'Peru', 'confirmed': 330123, 'dea...  07-14-2020  1594684800
5  {'location': 'Chile', 'confirmed': 317657, 'de...  07-14-2020  1594684800
6  {'location': 'Mexico', 'confirmed': 304435, 'd...  07-14-2020  1594684800
7  {'location': 'United Kingdom', 'confirmed': 29...  07-14-2020  1594684800
8  {'location': 'South Africa', 'confirmed': 2877...  07-14-2020  1594684800
9  {'location': 'Iran', 'confirmed': 259652, 'dea...  07-14-2020  1594684800
```

Head & Tail

In [ ]:

```
import pandas as pd

# Baca file sample_csv.csv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")

# Tampilkan 3 data teratas
print("Tiga data teratas:")
print(df.head(3))
```

```
# Tampilkan 3 data terbawah
print("Tiga data terbawah:")
print(df.tail(3))
```

```
Tiga data teratas:
   order_id  order_date  customer_id  ...    brand quantity item_price
0   1612339  2019-01-01        18055  ...  BRAND_C        4    1934000
1   1612339  2019-01-01        18055  ...  BRAND_V        8     604000
2   1612339  2019-01-01        18055  ...  BRAND_G       12     747000

[3 rows x 9 columns]
Tiga data terbawah:
     order_id  order_date  customer_id  ...    brand quantity item_price
98    1612390  2019-01-01        12681  ...  BRAND_S       24     450000
99    1612390  2019-01-01        12681  ...  BRAND_S       24     450000
100   1612390  2019-01-01        12681  ...  BRAND_B        4    1325000

[3 rows x 9 columns]
```

Indexing, Slicing, dan Transforming

Indexing - Part 2

In [ ]:

```
import pandas as pd

# Baca file TSV sample_tsv.tsv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_tsv.tsv",
sep="\t")

# Index dari df
print("Index:", df.index)

# Column dari df
print("Columns:", df.columns)
```

```
Index: RangeIndex(start=0, stop=101, step=1)
Columns: Index(['order_id', 'order_date', 'customer_id', 'city', 'province',
       'product_id', 'brand', 'quantity', 'item_price'],
      dtype='object')
```

Indexing - Part 3

In [ ]:

```
import pandas as pd

# Baca file TSV sample_tsv.tsv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_tsv.tsv",
sep="\t")

# Set multi index df
df_x = df.set_index(['order_date', 'city', 'customer_id'])

# Print nama dan level dari multi index
for name, level in zip(df_x.index.names, df_x.index.levels):
  print(name,':',level)
```

```
order_date : Index(['2019-01-01'], dtype='object', name='order_date')
city : Index(['Bogor', 'Jakarta Pusat', 'Jakarta Selatan', 'Jakarta Utara',
       'Makassar', 'Malang', 'Surabaya', 'Tangerang'],
      dtype='object', name='city')
customer_id : Int64Index([12681, 13963, 15649, 17091, 17228, 17450, 17470, 17511, 17616,
            18055],
           dtype='int64', name='customer_id')
```

Indexing - Part 4

Indexing - Part 4

In [ ]:

```
import pandas as pd

# Baca file sample_tsv.tsv untuk 10 baris pertama saja
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_tsv.tsv",
sep="\t", nrows=10)

# Cetak data frame awal
print("Dataframe awal:")
print(df)

# Set index baru
df.index = ["Pesanan ke-" + str(i) for i in range(1,11)]

# Cetak data frame dengan index baru
print("Dataframe dengan index baru:")
print(df)
```

```
Dataframe awal:
   order_id  order_date  customer_id  ...    brand quantity item_price
0   1612339  2019-01-01        18055  ...  BRAND_C        4    1934000
1   1612339  2019-01-01        18055  ...  BRAND_V        8     604000
2   1612339  2019-01-01        18055  ...  BRAND_G       12     747000
3   1612339  2019-01-01        18055  ...  BRAND_B       12     450000
4   1612339  2019-01-01        18055  ...  BRAND_G        3    1500000
5   1612339  2019-01-01        18055  ...  BRAND_V        3    2095000
6   1612339  2019-01-01        18055  ...  BRAND_H        3    2095000
7   1612339  2019-01-01        18055  ...  BRAND_S        3    1745000
8   1612339  2019-01-01        18055  ...  BRAND_F        6    1045000
9   1612339  2019-01-01        18055  ...  BRAND_P        6    1045000

[10 rows x 9 columns]
Dataframe dengan index baru:
               order_id  order_date  customer_id  ...    brand quantity item_price
Pesanan ke-1    1612339  2019-01-01        18055  ...  BRAND_C        4    1934000
Pesanan ke-2    1612339  2019-01-01        18055  ...  BRAND_V        8     604000
Pesanan ke-3    1612339  2019-01-01        18055  ...  BRAND_G       12     747000
Pesanan ke-4    1612339  2019-01-01        18055  ...  BRAND_B       12     450000
Pesanan ke-5    1612339  2019-01-01        18055  ...  BRAND_G        3    1500000
Pesanan ke-6    1612339  2019-01-01        18055  ...  BRAND_V        3    2095000
Pesanan ke-7    1612339  2019-01-01        18055  ...  BRAND_H        3    2095000
Pesanan ke-8    1612339  2019-01-01        18055  ...  BRAND_S        3    1745000
Pesanan ke-9    1612339  2019-01-01        18055  ...  BRAND_F        6    1045000
Pesanan ke-10   1612339  2019-01-01        18055  ...  BRAND_P        6    1045000

[10 rows x 9 columns]
```

Indexing Part - 5

In [ ]:

```
import pandas as pd

# Baca file sample_tsv.tsv dan set lah index_col sesuai instruksi
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_tsv.tsv",
sep="\t", index_col=["order_date", "order_id"])

# Cetak data frame untuk 8 data teratas
print("Dataframe:")
print(df.head(8))
```

```
Dataframe:
                     customer_id             city  ... quantity item_price
order_date order_id                                ...
2019-01-01 1612339         18055  Jakarta Selatan  ...        4    1934000
           1612339         18055  Jakarta Selatan  ...        8     604000
           1612339         18055  Jakarta Selatan  ...       12     747000
           1612339         18055  Jakarta Selatan  ...       12     450000
           1612339         18055  Jakarta Selatan  ...        3    1500000
```

```
1612339         18055  Jakarta Selatan  ...        3    2095000
1612339         18055  Jakarta Selatan  ...        3    2095000
1612339         18055  Jakarta Selatan  ...        3    1745000

[8 rows x 7 columns]
```

Slicing - Part 1

In [ ]:

```
import pandas as pd

# Baca file sample_csv.csv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")

# Slice langsung berdasarkan kolom
df_slice = df.loc[(df["customer_id"] == "18055") & (df["product_id"].isin(["P0029","P0040","P0041","P0116","P0117"]))]
print("Slice langsung berdasarkan kolom:")
print(df_slice)
```

```
Slice langsung berdasarkan kolom:
Empty DataFrame
Columns: [order_id, order_date, customer_id, city, province, product_id, brand, quantity, item_price]
Index: []
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/ops/array_ops.py:253: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  res_values = method(rvalues)
```

Sliicng - Part 2

In [ ]:

```
import pandas as pd

# Baca file sample_csv.csv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")

# Set index dari df sesuai instruksi
df = df.set_index(["order_date","order_id","product_id"])

# Slice sesuai intruksi
df_slice = df.loc[("2019-01-01",1612339,["P2154","P2159"]),:]
print("Slice df:")
print(df_slice)
```

```
Slice df:
                                customer_id  ... item_price
order_date order_id product_id               ...
2019-01-01 1612339  P2154             18055  ...    1745000
                    P2159             18055  ...     310000

[2 rows x 6 columns]
```

Transforming - Part 1

In [ ]:

```
import pandas as pd

# Baca file sample_csv.csv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")

# Tampilkan tipe data
print("Tipe data df:")
print(df.dtypes)
```

```
# Ubah tipe data kolom order_date menjadi datetime
df["order_date"] = pd.to_datetime(df["order_date"])

# Tampilkan tipe data df setelah transformasi
print("\nTipe data df setelah transformasi:")
print(df.dtypes)
```

```
Tipe data df:
order_id        int64
order_date     object
customer_id     int64
city           object
province       object
product_id     object
brand          object
quantity        int64
item_price      int64
dtype: object

Tipe data df setelah transformasi:
order_id                int64
order_date     datetime64[ns]
customer_id             int64
city                   object
province               object
product_id             object
brand                  object
quantity                int64
item_price              int64
dtype: object
```

Transforming - Part 2

In [ ]:

```
import pandas as pd

# Baca file sample_csv.csv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")

# Tampilkan tipe data
print("Tipe data df:")
print(df.dtypes)

# Ubah tipe data kolom quantity menjadi tipe data numerik float
df["quantity"] = pd.to_numeric(df["quantity"], downcast="float")

# Ubah tipe data kolom city menjadi tipe data category
df["city"] = df["city"].astype("category")

# Tampilkan tipe data df setelah transformasi
print("\nTipe data df setelah transformasi:")
print(df.dtypes)
```

```
Tipe data df:
order_id        int64
order_date     object
customer_id     int64
city           object
province       object
product_id     object
brand          object
quantity        int64
item_price      int64
dtype: object

Tipe data df setelah transformasi:
order_id          int64
order_date       object
customer_id       int64
city           category
```

```
city            category
province          object
product_id        object
brand             object
quantity         float32
item_price         int64
dtype: object
```

Transforming - Part 3

In [ ]:

```
import pandas as pd

# Baca file sample_csv.csv
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/sample_csv.csv")

# Cetak 5 baris teratas kolom brand
print("Kolom brand awal:")
print(df["brand"].head())

# Gunakan method apply untuk merubah isi kolom menjadi lower case
df["brand"] = df["brand"].apply(lambda x: x.lower())

# Cetak 5 baris teratas kolom brand
print("Kolom brand setelah apply:")
print(df["brand"].head())

# Gunakan method map untuk mengambil kode brand yaitu karakter terakhirnya
df["brand"] = df["brand"].map(lambda x: x[-1])

# Cetak 5 baris teratas kolom brand
print("Kolom brand setelah map:")
print(df["brand"].head())
```

```
Kolom brand awal:
0    BRAND_C
1    BRAND_V
2    BRAND_G
3    BRAND_B
4    BRAND_G
Name: brand, dtype: object
Kolom brand setelah apply:
0    brand_c
1    brand_v
2    brand_g
3    brand_b
4    brand_g
Name: brand, dtype: object
Kolom brand setelah map:
0    c
1    v
2    g
3    b
4    g
Name: brand, dtype: object
```

Transforming - Part 4

In [ ]:

```
import numpy as np
import pandas as pd

# number generator, set angka seed menjadi suatu angka, bisa semua angka, supaya hasil ra
ndom nya selalu sama ketika kita run
np.random.seed(1234)

# create dataframe 3 baris dan 4 kolom dengan angka random
df_tr = pd.DataFrame(np.random.rand(3,4))
```

```
# Cetak dataframe
print("Dataframe:")
print(df_tr)

# Cara 1 dengan tanpa define function awalnya, langsung pake fungsi anonymous lambda x
df_tr1 = df_tr.applymap(lambda x: x**2 + 3*x + 2)
print("\nDataframe - cara 1:")
print(df_tr1)

# Cara 2 dengan define function
def qudratic_fun(x):
 return x**2 + 3*x + 2
df_tr2 = df_tr.applymap(qudratic_fun)
print("\nDataframe - cara 2:")
print(df_tr2)
```

```
Dataframe:
          0         1         2         3
0  0.191519  0.622109  0.437728  0.785359
1  0.779976  0.272593  0.276464  0.801872
2  0.958139  0.875933  0.357817  0.500995

Dataframe - cara 1:
          0         1         2         3
0  2.611238  4.253346  3.504789  4.972864
1  4.948290  2.892085  2.905825  5.048616
2  5.792449  5.395056  3.201485  3.753981

Dataframe - cara 2:
          0         1         2         3
0  2.611238  4.253346  3.504789  4.972864
1  4.948290  2.892085  2.905825  5.048616
2  5.792449  5.395056  3.201485  3.753981
```

Handling Missing Values

Inspeksi Missing Value

In [ ]:

```
import pandas as pd

# Baca file "public data covid19 jhu csse eu.csv"
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/CHAPTER+4+-+missi
ng+value+-+public+data+covid19+.csv")

# Cetak info dari df
print(df.info())

# Cetak jumlah missing value di setiap kolom
mv = df.isna().sum()
print("\nJumlah missing value per kolom:")
print(mv)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   province_state  960 non-null    object
 1   country_region  1000 non-null   object
 2   date            1000 non-null   object
 3   latitude        874 non-null    float64
 4   longitude       874 non-null    float64
 5   location_geom   874 non-null    object
 6   confirmed       1000 non-null   int64
 7   deaths          999 non-null    float64
 8   recovered       999 non-null    float64
 9   active          949 non-null    float64
 10  fips            949 non-null    float64
```

```
 11  admin2          842 non-null    object
 12  combined_key    0 non-null      float64
dtypes: float64(7), int64(1), object(5)
memory usage: 101.7+ KB
None

Jumlah missing value per kolom:
province_state      40
country_region       0
date                 0
latitude           126
longitude          126
location_geom      126
confirmed            0
deaths               1
recovered            1
active              51
fips                51
admin2             158
combined_key      1000
dtype: int64
```

Treatment untuk Missing Value - Part 2

In [ ]:

```
import pandas as pd

# Baca file "public data covid19 jhu csse eu.csv"
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/CHAPTER+4+-+missing+value+-+public+data+covid19+.csv")

# Cetak ukuran awal dataframe
print("Ukuran awal df: %d baris, %d kolom." % df.shape)

# Drop kolom yang seluruhnya missing value dan cetak ukurannya
df = df.dropna(axis=1, how="all")
print("Ukuran df setelah buang kolom dengan seluruh data missing: %d baris, %d kolom." % df.shape)

# Drop baris jika ada satu saja data yang missing dan cetak ukurannya
df = df.dropna(axis=0, how="any")
print("Ukuran df setelah dibuang baris yang memiliki sekurangnya 1 missing value: %d baris, %d kolom." % df.shape)
```

```
Ukuran awal df: 1000 baris, 13 kolom.
Ukuran df setelah buang kolom dengan seluruh data missing: 1000 baris, 12 kolom.
Ukuran df setelah dibuang baris yang memiliki sekurangnya 1 missing value: 746 baris, 12 kolom.
```

Treatment untuk Missing Value - Part 3

In [ ]:

```
import pandas as pd

# Baca file "public data covid19 jhu csse eu.csv"
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/CHAPTER+4+-+missing+value+-+public+data+covid19+.csv")

# Cetak unique value pada kolom province_state
print("Unique value awal:")
print(df["province_state"].unique())

# Ganti missing value dengan string "unknown_province_state"
df["province_state"] = df["province_state"].fillna("unknown_province_state")

# Cetak kembali unique value pada kolom province_state
print("Unique value setelah fillna:")
print(df["province_state"].unique())
```

```
Unique value awal:
[nan 'US' 'Guam' 'Iowa']
Unique value setelah fillna:
['unknown_province_state' 'US' 'Guam' 'Iowa']
```

Treatment untuk Missing Value - Part 4

In [ ]:

```
import pandas as pd

# Baca file "https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/CHAPTER+4+-+missing+value+-+public+data+covid19+.csv"
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/CHAPTER+4+-+missing+value+-+public+data+covid19+.csv")

# Cetak nilai mean dan median awal
print("Awal: mean = %f, median = %f." % (df["active"].mean(), df["active"].median()))

# Isi missing value kolom active dengan median
df_median = df["active"].fillna(df["active"].median())

# Cetak nilai mean dan median awal setelah diisi dengan median
print("Fillna median: mean = %f, median = %f." % (df_median.mean(), df_median.median()))

# Isi missing value kolom active dengan mean
df_mean = df["active"].fillna(df["active"].mean())

# Cetak nilai mean dan median awal setelah diisi dengan mean
print("Fillna mean: mean = %f, median = %f." % (df_mean.mean(), df_mean.median()))
```

```
Awal: mean = 192.571128, median = 41.000000.
Fillna median: mean = 184.841000, median = 41.000000.
Fillna mean: mean = 192.571128, median = 49.000000.
```

Treatment untuk Missing Value - Part 5

In [ ]:

```
import numpy as np
import pandas as pd

# Data
ts = pd.Series({
   "2020-01-01":9,
   "2020-01-02":np.nan,
   "2020-01-05":np.nan,
   "2020-01-07":24,
   "2020-01-10":np.nan,
   "2020-01-12":np.nan,
   "2020-01-15":33,
   "2020-01-17":np.nan,
   "2020-01-16":40,
   "2020-01-20":45,
   "2020-01-22":52,
   "2020-01-25":75,
   "2020-01-28":np.nan,
   "2020-01-30":np.nan
})

# Isi missing value menggunakan interpolasi linier
ts = ts.interpolate()

# Cetak time series setelah interpolasi linier
print("Setelah diisi missing valuenya:")
print(ts)
```

```
Setelah diisi missing valuenya:
2020-01-01     9.0
2020-01-02    14.0
2020-01-05    19.0
```

```
2020-01-07    24.0
2020-01-10    27.0
2020-01-12    30.0
2020-01-15    33.0
2020-01-17    36.5
2020-01-16    40.0
2020-01-20    45.0
2020-01-22    52.0
2020-01-25    75.0
2020-01-28    75.0
2020-01-30    75.0
dtype: float64
```

Mini Project

In [ ]:

```
import pandas as pd

# 1. Baca dataset
print("[1] BACA DATASET")
df = pd.read_csv("https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/retail_raw_test.csv", low_memory=False)
print("    Dataset:\n", df.head())
print("    Info:\n", df.info())

# 2. Ubah tipe data
print("\n[2] UBAH TIPE DATA")
df["customer_id"] = df["customer_id"].apply(lambda x: x.split("'")[1]).astype("int64")
df["quantity"] = df["quantity"].apply(lambda x: x.split("'")[1]).astype("int64")
df["item_price"] = df["item_price"].apply(lambda x: x.split("'")[1]).astype("int64")
print("    Tipe data:\n", df.dtypes)

# 3. Transform "product_value" supaya bentuknya seragam dengan format "PXXXX", assign ke kolom baru "product_id", dan drop kolom "product_value", jika terdapat nan gantilah dengan "unknown"
print("\n[3] TRANSFORM product_value MENJADI product_id")
# Buat fungsi
import math
def impute_product_value(val):
    if math.isnan(val):
      return "unknown"
    else:
      return 'P' + '{:0>4}'.format(str(val).split('.')[0])
# Buat kolom "product_id"
df["product_id"] = df["product_value"].apply(lambda x: impute_product_value(x))
# Hapus kolom "product_value"
df.drop(["product_value"], axis=1, inplace=True)
# Cetak 5 data teratas
print(df.head())

# 4. Tranform order_date menjadi value dengan format "YYYY-mm-dd"
print("\n[4] TRANSFORM order_date MENJADI FORMAT YYYY-mm-dd")
months_dict = {
   "Jan":"01",
   "Feb":"02",
   "Mar":"03",
   "Apr":"04",
   "May":"05",
   "Jun":"06",
   "Jul":"07",
   "Aug":"08",
   "Sep":"09",
   "Oct":"10",
   "Nov":"11",
   "Dec":"12"
}
df["order_date"] = pd.to_datetime(df["order_date"].apply(lambda x: str(x)[-4:] + "-" + months_dict[str(x)[:3]] + "-" + str(x)[4:7]))
print("    Tipe data:\n", df.dtypes)
```

```python
# 5. Mengatasi data yang hilang di beberapa kolom
print("\n[5] HANDLING MISSING VALUE")
# Kolom "city" dan "province" masih memiliki missing value, nilai yang hilang di kedua ko
lom ini diisi saja dengan "unknown"
df[["city","province"]] = df[["city","province"]].fillna("unknown")
# Kolom brand juga masih memiliki missing value, Ganti value NaN menjadi "no_brand"
df["brand"] = df["brand"].fillna("no_brand")
# Cek apakah masih terdapat missing value di seluruh kolom
print("    Info:\n", df.info())

# 6. Membuat kolom baru "city/province" dengan menggabungkan kolom "city" dan kolom "prov
ince" dan delete kolom asalnya
print("\n[6] MEMBUAT KOLOM BARU city/province")
df["city/province"] = df["city"] + "/" + df["province"]
# drop kolom "city" dan "province" karena telah digabungkan
df.drop(["city","province"], axis=1, inplace=True)
# Cetak 5 data teratas
print(df.head())

# 7. Membuat hierarchical index yang terdiri dari kolom "city/province", "order_date", "c
ustomer_id", "order_id", "product_id"
print("\n[7] MEMBUAT HIERACHICAL INDEX")
df = df.set_index(["city/province","order_date","customer_id","order_id","product_id"])
# urutkanlah berdasarkan index yang baru
df = df.sort_index()
# Cetak 5 data teratas
print(df.head())

# 8. Membuat kolom "total_price" yang formula nya perkalian antara kolom "quantity" dan k
olom "item_price"
print("\n[8] MEMBUAT KOLOM total_price")
df["total_price"] = df["quantity"] * df["item_price"]
# Cetak 5 data teratas
print(df.head())

# 9. Slice dataset agar hanya terdapat data bulan Januari 2019
print("\n[9] SLICE DATASET UNTUK BULAN JANUARI 2019 SAJA")
idx = pd.IndexSlice
df_jan2019 = df.loc[idx[:, "2019-01-01":"2019-01-31"], :]
print("Dataset akhir:\n", df_jan2019)

# END OF PROJECT
```

```
[1] BACA DATASET
    Dataset:
    order_id    order_date customer_id  ... quantity item_price product_value
0   1730350  Dec 11, 2019      '13447  ...      '24    '113000        1374.0
1   1677490  Jul 31, 2019          '0  ...       '1   '1164000        1370.0
2   1704211  Oct 18, 2019      '16128  ...      '12    '747000        1679.0
3   1679695  Aug 07, 2019      '16225  ...       '6    '590000        1708.0
4   1679080  Aug 05, 2019          '0  ...       '2    '740000        1201.0

[5 rows x 9 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   order_id       5000 non-null   int64
 1   order_date     5000 non-null   object
 2   customer_id    5000 non-null   object
 3   city           3802 non-null   object
 4   province       3802 non-null   object
 5   brand          4995 non-null   object
 6   quantity       5000 non-null   object
 7   item_price     5000 non-null   object
 8   product_value  4995 non-null   float64
dtypes: float64(1), int64(1), object(7)
memory usage: 351.7+ KB
    Info:
 None
```

```
[2] UBAH TIPE DATA
    Tipe data:
 order_id           int64
order_date        object
customer_id        int64
city              object
province          object
brand             object
quantity           int64
item_price         int64
product_value    float64
dtype: object

[3] TRANSFORM product_value MENJADI product_id
   order_id    order_date  customer_id  ... quantity item_price product_id
0   1730350  Dec 11, 2019        13447  ...       24     113000      P1374
1   1677490  Jul 31, 2019            0  ...        1    1164000      P1370
2   1704211  Oct 18, 2019        16128  ...       12     747000      P1679
3   1679695  Aug 07, 2019        16225  ...        6     590000      P1708
4   1679080  Aug 05, 2019            0  ...        2     740000      P1201

[5 rows x 9 columns]

[4] TRANSFORM order_date MENJADI FORMAT YYYY-mm-dd
    Tipe data:
 order_id               int64
order_date     datetime64[ns]
customer_id             int64
city                   object
province               object
brand                  object
quantity                int64
item_price              int64
product_id             object
dtype: object

[5] HANDLING MISSING VALUE
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   order_id     5000 non-null   int64
 1   order_date   5000 non-null   datetime64[ns]
 2   customer_id  5000 non-null   int64
 3   city         5000 non-null   object
 4   province     5000 non-null   object
 5   brand        5000 non-null   object
 6   quantity     5000 non-null   int64
 7   item_price   5000 non-null   int64
 8   product_id   5000 non-null   object
dtypes: datetime64[ns](1), int64(4), object(4)
memory usage: 351.7+ KB
    Info:
 None

[6] MEMBUAT KOLOM BARU city/province
   order_id order_date  ...  product_id              city/province
0   1730350 2019-12-11  ...       P1374      Surakarta/Jawa Tengah
1   1677490 2019-07-31  ...       P1370            unknown/unknown
2   1704211 2019-10-18  ...       P1679  Jakarta Pusat/DKI Jakarta
3   1679695 2019-08-07  ...       P1708      Yogyakarta/Yogyakarta
4   1679080 2019-08-05  ...       P1201            unknown/unknown

[5 rows x 8 columns]

[7] MEMBUAT HIERACHICAL INDEX
                                                                  brand  ...  item_p
rice
city/province          order_date customer_id order_id product_id           ...
```

```
Banda Aceh/Aceh         2019-04-17 12818       1642480  P1936       BRAND_K  ...      450
000
                        2019-11-12 12360       1715116  P0758       BRAND_C  ...      695
000
                                                        P3042       BRAND_R  ...       31
0000
                        2019-12-09 12374       1729036  P1660       BRAND_G  ...     2795
000
Bandar Lampung/Lampung  2019-01-15 12515       1619257  P0628       BRAND_C  ...     6950
00

[5 rows x 3 columns]

[8] MEMBUAT KOLOM total_price
                                                                      brand  ...  total_
price
city/province           order_date customer_id order_id product_id           ...

Banda Aceh/Aceh         2019-04-17 12818       1642480  P1936       BRAND_K  ...     1080
0000
                        2019-11-12 12360       1715116  P0758       BRAND_C  ...      556
0000
                                                        P3042       BRAND_R  ...       37
20000
                        2019-12-09 12374       1729036  P1660       BRAND_G  ...     1118
0000
Bandar Lampung/Lampung  2019-01-15 12515       1619257  P0628       BRAND_C  ...     8340
000

[5 rows x 4 columns]

[9] SLICE DATASET UNTUK BULAN JANUARI 2019 SAJA
Dataset akhir:
                                                                       brand  ...  total
_price
city/province           order_date customer_id order_id product_id           ...

Bandar Lampung/Lampung  2019-01-15 12515       1619257  P0628       BRAND_C  ...     8340
000
Bandung/Jawa Barat      2019-01-09 16134       1617055  P1597       BRAND_G  ...      468
0000
                        2019-01-10 17392       1617952  P2137       BRAND_M  ...      212
4000
                        2019-01-14 15527       1618828  P3115       BRAND_S  ...      104
5000
                        2019-01-29 13253       1620289  P0099       BRAND_A  ...      540
0000
...                                                                     ...  ...
...
unknown/unknown         2019-01-30 0           1620766  P3070       BRAND_R  ...       59
3000
                                                        P3483       BRAND_S  ...       17
79000
                        2019-01-31 0           1621057  P1298       BRAND_F  ...       29
6000
                                                        P1773       BRAND_H  ...       29
65000
                                                        P2877       BRAND_R  ...       14
86000

[334 rows x 4 columns]
```

Kesimpulan

1. **Memahami library Pandas dan interaksinya dengan numpy**
2. **Memahami dan mempraktekkkan bagaimana membuat series dan dataframe pada pandas dari berbagai tipe data seperti list, list of list, dict, ataupun numpy array**
3. **Memahami dan mempraktekkkan bagaimana membaca dataset dari berbagai format standar seperti csv, tsc, excel, json, sql sehingga dapat dijadikan pandas dataframe/series serta bagaimana cara menyimpannya ke format standar dataset.**

4. **Mampu memahami dan mempraktekkan proses indexing, transformasi dan slicing pada dataframe**
5. **Mampu memahami dan mempraktekkan bagaimana cara menghandle missing value pada suatu dataframe**
6. **Latihan dalam mengerjakan project bisnis sederhana menggunakan pandas**