Python for Data Analysis: Data Preprocessing with Pandas

MInggu, 2 April 2023

Arif Romadhan
Sr. Data Scientist

Program Zenius Studi Independen
Bersertifikat Bersama Kampus Merdeka

PT Cakra Syntesis Indonesia
Android Developer (2016)

Nawatech
Jr. ML Engineer (2017)

Bukalapak
Data Scientist (2018 - 2020)

Data Scientist Instructor
(Sept 2020 - Nov2020)

Data Scientist Instructor
(2021 - 2021)

Evermos
Sr. Data Scientist - Data Lead
(2021 - present)

Data Scientist Instructor
(2021- present)

Arif Romadhan

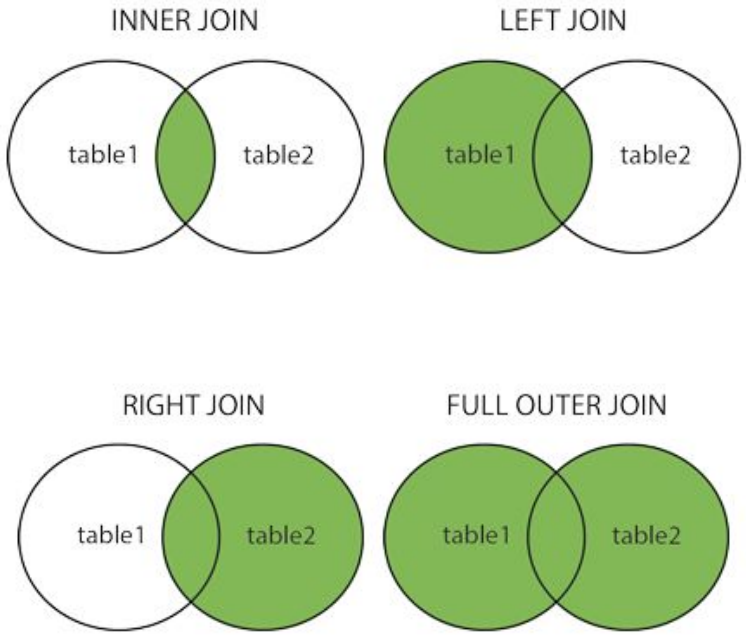
https://www.linkedin.com/in/arif-romadhan19/

1. Pandas Join
2. Data Aggregation
3. Apply Function
4. Datetime Handling

Joins

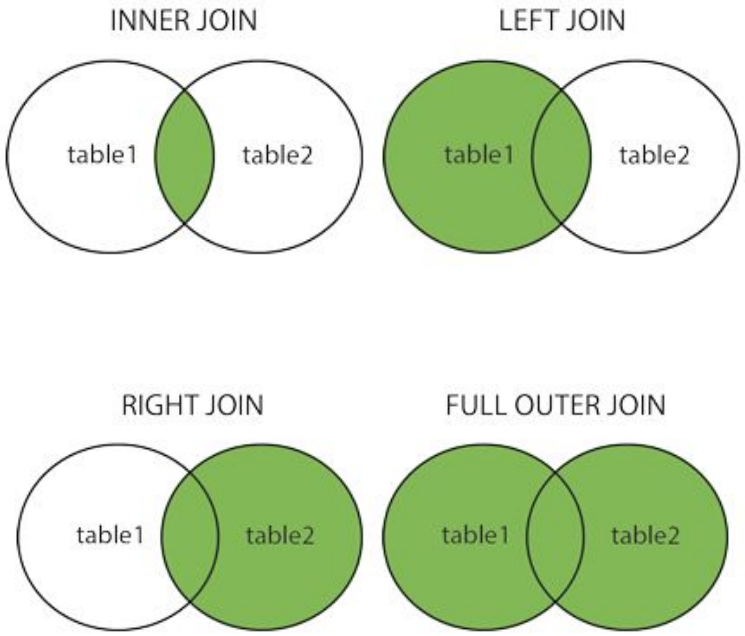
What is Join?

The process of combining two columns from separate table.

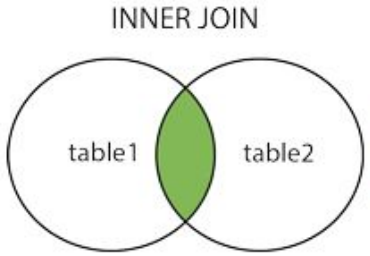


Type of Join

Type	Description
INNER JOIN	matching values in both tables
LEFT JOIN	all records from the left table, and the matched records from the right table
RIGHT JOIN	all records from the right table, and the matched records from the left table
FULL OUTER JOIN	all records when there is a match in either left or right table



Inner Join




```
pd.merge(df1,df2,on='id')
```

matching values in both tables

Table 1

id	Name
1	Budi
2	Yosua
3	Adel
4	Dinda

Table 2

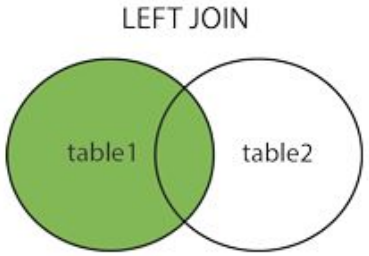
id	age
1	20
2	30
3	40
5	20
6	25

inner join by id

Output

id	Name	age
1	Budi	20
2	Yosua	30
3	Adel	40

Left Join




```
pd.merge(df1,df2,on='id',how='left')
```

all records from the left table, and the matched records from the right table

Table 1

id	Name
1	Budi
2	Yosua
3	Adel
4	Dinda

Table 2

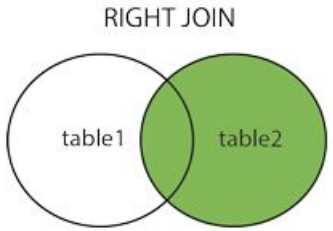
id	age
1	20
2	30
3	40
5	20
6	25

left join by id

Output

id	Name	age
1	Budi	20.0
2	Yosua	30.0
3	Adel	40.0
4	Dinda	NaN

Right Join




```
pd.merge(df1,df2,on='id',how='right')
```

all records from the right table, and the matched records from the left table

Table 1

id	Name
1	Budi
2	Yosua
3	Adel
4	Dinda

Table 2

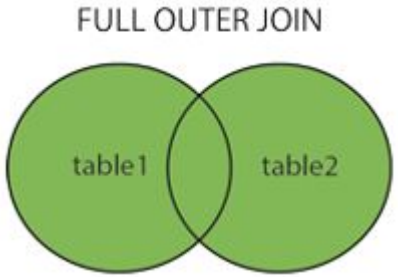
id	age
1	20
2	30
3	40
5	20
6	25

right join by id

Output

id	Name	age
1	Budi	20
2	Yosua	30
3	Adel	40
5	NaN	20
6	NaN	25

Outer Join




```
pd.merge(df1,df2,on='id',how='outer')
```

all records when there is a match in either left or right table

Table 1

id	Name
1	Budi
2	Yosua
3	Adel
4	Dinda

Table 2

id	age
1	20
2	30
3	40
5	20
6	25

outer join by id →

Output

id	Name	age
1	Budi	20.0
2	Yosua	30.0
3	Adel	40.0
4	Dinda	NaN
5	NaN	20.0
6	NaN	25.0

Appending Dataframe

Appending Dictionary to Dataframe

Append to df by using dictionary

	Product	Price	Stock
0	Laptop	2000	10
1	Tablet	800	30
2	Mobile	3000	20

```
dictionary = {
    'Product':'PC',
    'Price':3000,
    'Stock':5
}
```

```
df.append(dictionary, ignore_index=True)
```

	Product	Price	Stock
0	Laptop	2000	10
1	Tablet	800	30
2	Mobile	3000	20
3	PC	3000	5

Appending List to Dataframe

Append to df by using list

	Product	Price	Stock
0	Laptop	2000	10
1	Tablet	800	30
2	Mobile	3000	20

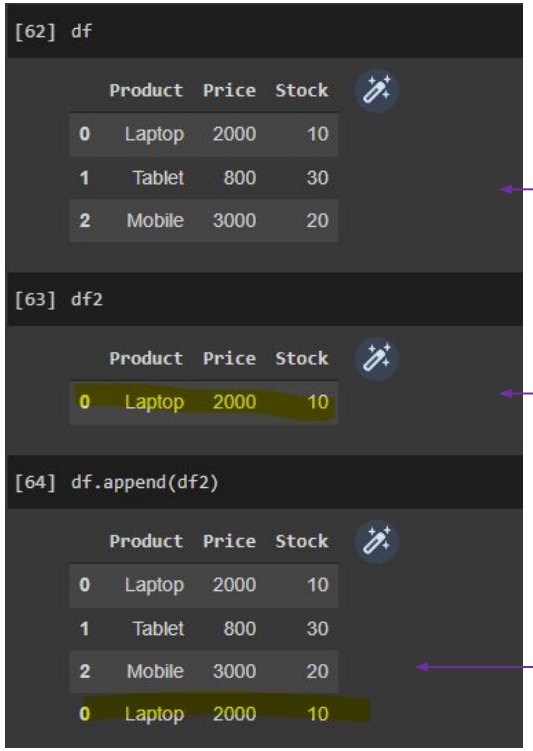
```
list_=["PC",3000,5]

df_length = len(df)
df.loc[df_length] = list
df
```

	Product	Price	Stock
0	Laptop	2000	10
1	Tablet	800	30
2	Mobile	3000	20
3	PC	3000	5

Appending Dataframe to Dataframe

Append dataframe to another dataframe



```
[62] df
```

	Product	Price	Stock
0	Laptop	2000	10
1	Tablet	800	30
2	Mobile	3000	20

```
[63] df2
```

	Product	Price	Stock
0	Laptop	2000	10

```
[64] df.append(df2)
```

	Product	Price	Stock
0	Laptop	2000	10
1	Tablet	800	30
2	Mobile	3000	20
0	Laptop	2000	10

Renaming Column

Renaming Column

We can rename column by using this syntax

```
df = df.rename(columns  = {
    `old_name_1' : `new_name_1',
    `old_name_2' : `new_name_2',
} )
```

Deleting Column

Deleting Column

We can delete column by using this syntax

```
df = df.drop('column_name', axis=1)
```

Filtering Data

Filtering Data

Below are several syntax we can apply to filter the data

Syntax	Function
df[df[`column’] == a ] df[ df[`column’] > a] df[df[`column’] < a]	Filter for specific value based on the condition
df[df[`column’].isin([a, b, c])]	Filter for multiple specific value
df[df[`column’].notnull()]	Filter out null value
df[df[`column’].str.contains(`string’)]	Filter for row which contains the string
df[df[`column’].str.like(`string%’)]	Filter for row with wildcard (%)

Sort Value

Sort Value

Below are several syntax we can use to sort the value on dataframe

```
df.sort_values('Year_Birth', ascending=False)
```

Data Aggregation

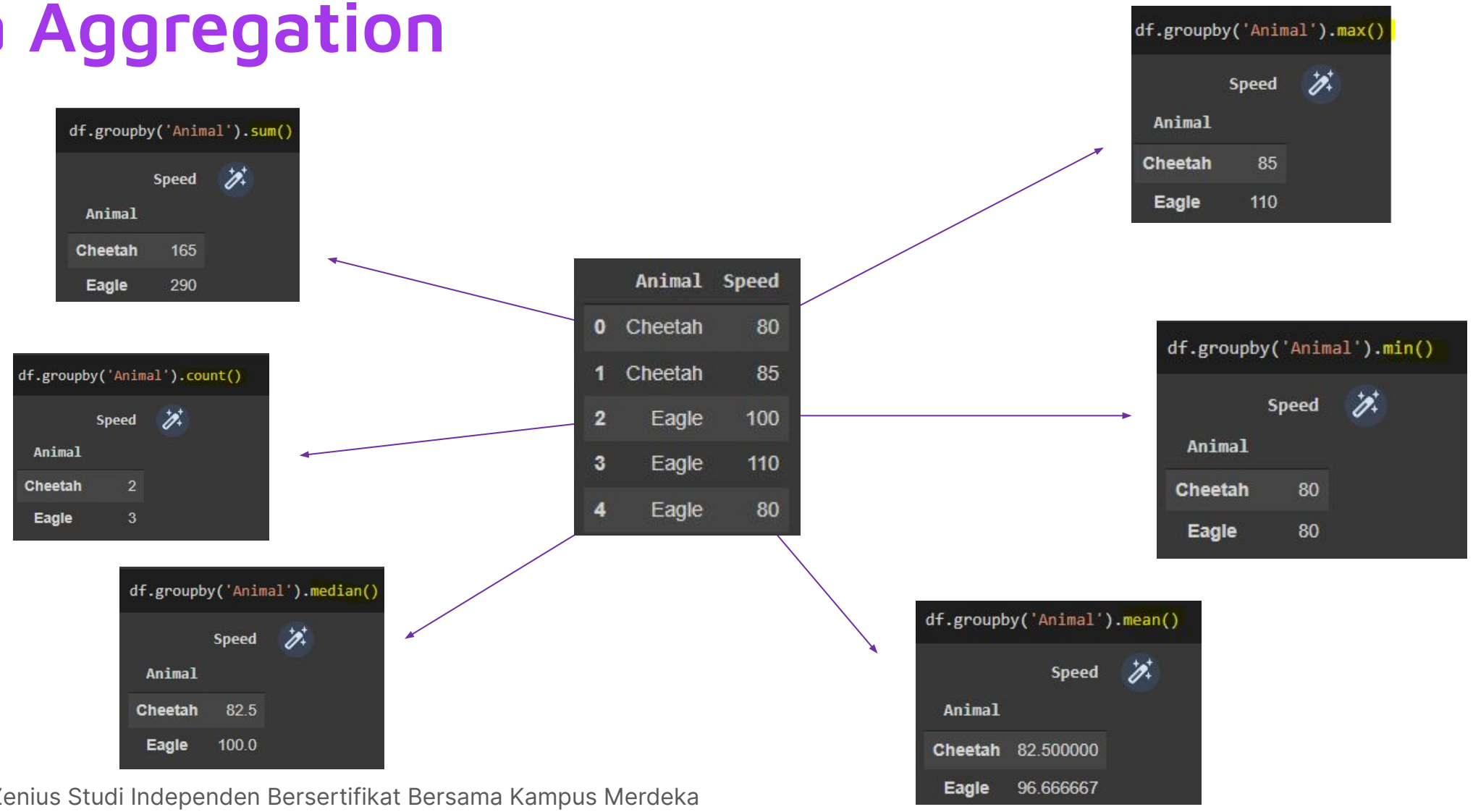
Data Aggregation

Data aggregation is the process where raw data is gathered and expressed in a summary.

Data can be aggregated over a given time period to provide statistics such as average, minimum, maximum, sum, and count

In pandas, Aggregation can be done by using “Groupby” method

Data Aggregation



Multiple Aggregation

	Animal	Speed
0	Cheetah	80
1	Cheetah	85
2	Eagle	100
3	Eagle	110
4	Eagle	80


```
df.groupby('Animal').agg({'Speed': ['mean', 'min', 'max']})
```

	Speed		
	mean	min	max
Animal			
Cheetah	82.500000	80	85
Eagle	96.666667	80	110

Pandas Function

Pandas Map Function

Map function will substitute each value in a Series with another value

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
df['Survived']=df.Survived.map({0:'Not Survived',1:'Survived'})
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	Not Survived	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	Survived	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	Survived	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	Survived	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	Not Survived	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Pandas Apply Function

Apply a function along an axis of the DataFrame.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	Not Survived	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	Survived	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	Survived	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	Survived	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	Not Survived	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
df['name_length']=df.Name.apply(len)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	name_length
0	1	Not Survived	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	23
1	2	Survived	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	51
2	3	Survived	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	22
3	4	Survived	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	44
4	5	Not Survived	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	24

In this figure, we are getting the length of the name and create a column out from it

Pandas Apply Function

Apply a function from library.

```
[57] df['name_length']=df.Name.apply(len)
     df[['Fare']].head()
```

	Fare
0	7.2500
1	71.2833
2	7.9250
3	53.1000
4	8.0500

```
[56] import numpy as np
     df['Round_Up_Fare']=df.Fare.apply(np.ceil)
     df[['Fare','Round_Up_Fare']].head()
```

	Fare	Round_Up_Fare
0	7.2500	8.0
1	71.2833	72.0
2	7.9250	8.0
3	53.1000	54.0
4	8.0500	9.0

In this figure, we are getting the ceil function from numpy library

Pandas Date Processing

Datetime Datatype

In order to process the datetime data in Dataframe. We need to convert it into Datetime type.

df['column_name']=pd.to_datetime(df['column_name'])

	City	Colors Reported	Shape Reported	State	Time
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00
2	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	Abilene	NaN	DISK	KS	6/1/1931 13:00
4	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00
...	...	...	...	...	...
18236	Grant Park	NaN	TRIANGLE	IL	12/31/2000 23:00
18237	Spirit Lake	NaN	DISK	IA	12/31/2000 23:00
18238	Eagle River	NaN	NaN	WI	12/31/2000 23:45
18239	Eagle River	RED	LIGHT	WI	12/31/2000 23:45
18240	Ybor	NaN	OVAL	FL	12/31/2000 23:59

18241 rows × 5 columns

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18241 entries, 0 to 18240
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   City             18216 non-null  object
 1   Colors Reported  2882 non-null   object
 2   Shape Reported   15597 non-null  object
 3   State            18241 non-null  object
 4   Time             18241 non-null  object
dtypes: object(5)
```

```
df['Time'] =  pd.to_datetime(df['Time'])
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18241 entries, 0 to 18240
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   City             18216 non-null  object
 1   Colors Reported  2882 non-null   object
 2   Shape Reported   15597 non-null  object
 3   State            18241 non-null  object
 4   Time             18241 non-null  datetime64[ns]
dtypes: datetime64[ns](1), object(4)
memory usage: 712.7+ KB
```

Get Properties

You can get specific properties from a Datetime data type i.e. (Hour,Week,Day,Month)

Get day of the year
df.column_name.dt.dayofyear

Get month
*df.column_name.dt.*month

Get day
*df.column_name.dt.*day

Get year
*df.column_name.dt.*year

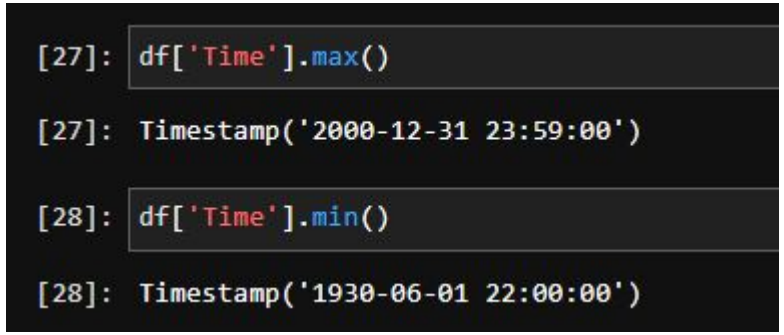
```
df['Time'] =  pd.to_datetime(df['Time'])
df.Time.dt.month
```

```
0         6
1         6
2         2
3         6
4         4
         ..
18236    12
18237    12
18238    12
18239    12
18240    12
Name: Time, Length: 18241, dtype: int64
```

For more properties:
https://pandas.pydata.org/docs/user_guide/timeseries.html#time-date-components

Max & Min

We can use Max to get the **latest date** and Min to get **the earliest date.**



Get Data between Date Range

We can use conditional clause on date time data. Refer to the image below.

```
condition= df['Time']>=pd.to_datetime('1950-1-1')
df.loc[condition].head()
```

	City	Colors Reported	Shape Reported	State	Time
118	Mount Hope	NaN	NaN	WV	1950-01-01 22:00:00
119	New York City	NaN	OVAL	NY	1950-01-02 00:00:00
120	Roswell	RED	NaN	NM	1950-03-22 00:00:00
121	Arkansas	NaN	DISK	AR	1950-04-15 08:00:00
122	Waynesborough	NaN	DISK	VA	1950-04-15 14:00:00

Return all rows after the first of January 1950

Assignment

Assignment 3

Instruksi Assignment 3A

Lakukan beberapa langkah berikut pada dataset **telco churn**

- Missing Values Checking
- Categorical Data Encoding
- Anomalies and Outlier Handling

Buatlah slide **presentasi** yang memuat penjelasan terkait langkah-langkah yang sudah ditempuh dan temuan yang didapatkan. Sertakan link **Google Colab** pada slide presentasi tersebut.

Instruksi Assignment 3B

Kerjakan soal-soal pada notebook yang terdapat pada

Topic 5 & 6 - Assignment | Hands-On Python 2 di Canvas.

Beri nama notebook yang kamu kerjakan dengan format: **Topik 6 - [Nama Lengkap].ipynb**. Sertakan link **Google Colab** dari notebook yang kamu kerjakan pada slide presentasi.

Kumpulkan file slide presentasi dalam format PDF dengan format nama file: **Topic 5 6 - [Nama Lengkap].pdf** dan sertakan link google colab pada pdf tersebut

Available from	Until
Apr 02 at 03.00 PM	Apr 10 at 11.59PM

Thank you!

Any Questions?