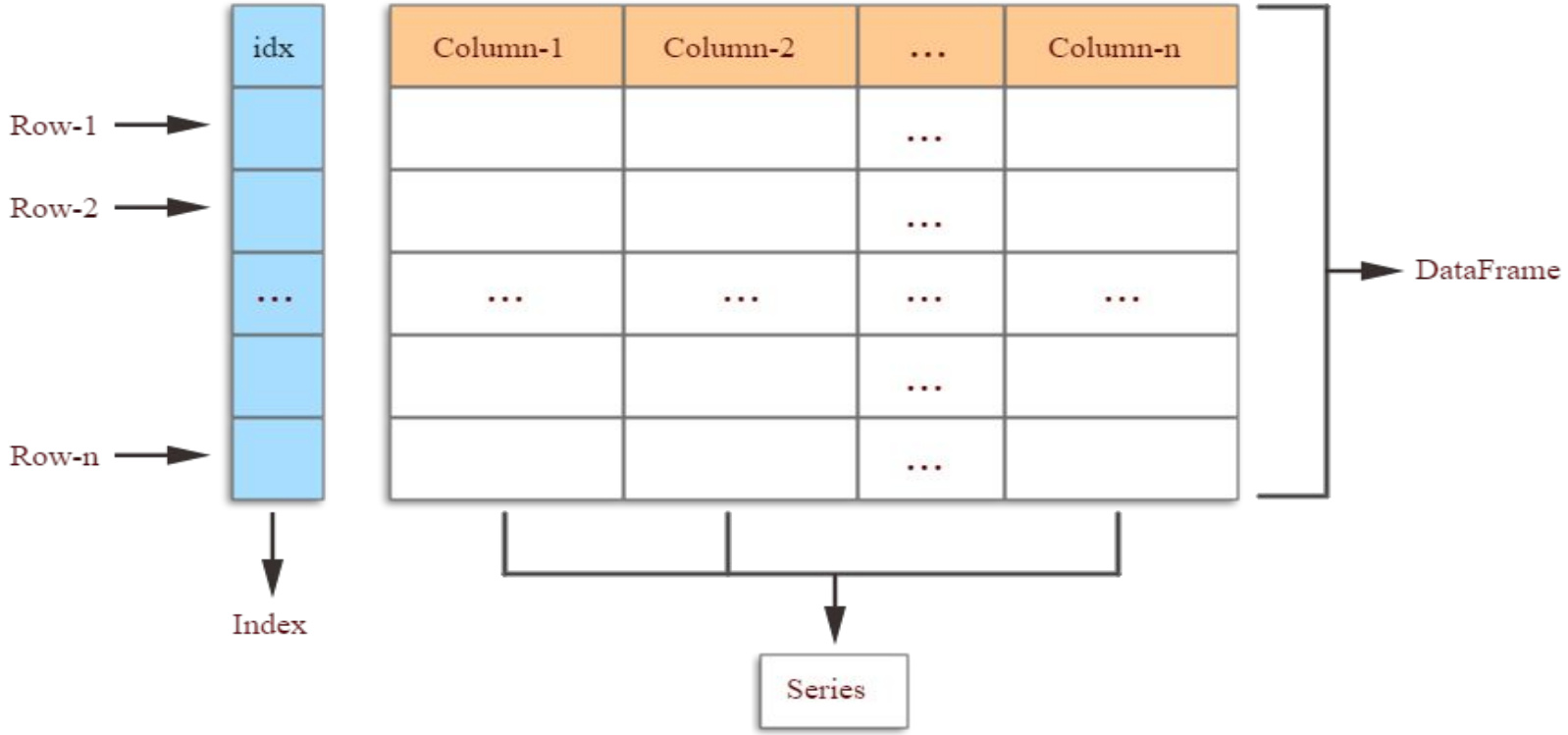
Introduction to Pandas

Explanation: Pandas is a Python library for data manipulation and analysis. It provides two main data structures: **Series and DataFrame**, which are designed to handle one-dimensional and two-dimensional data, respectively. Pandas is widely used in data science and is an essential tool for any data analysis project.

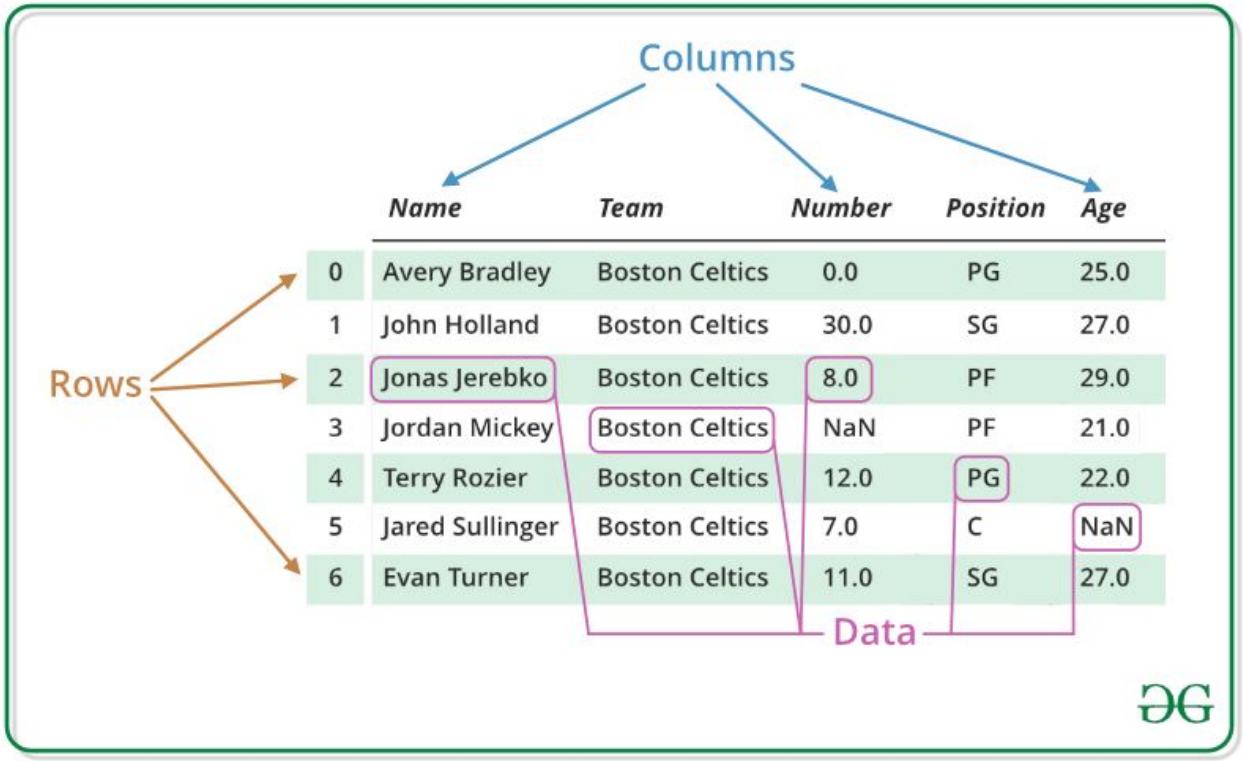
Install Pandas: **pip install pandas**

Pandas Features

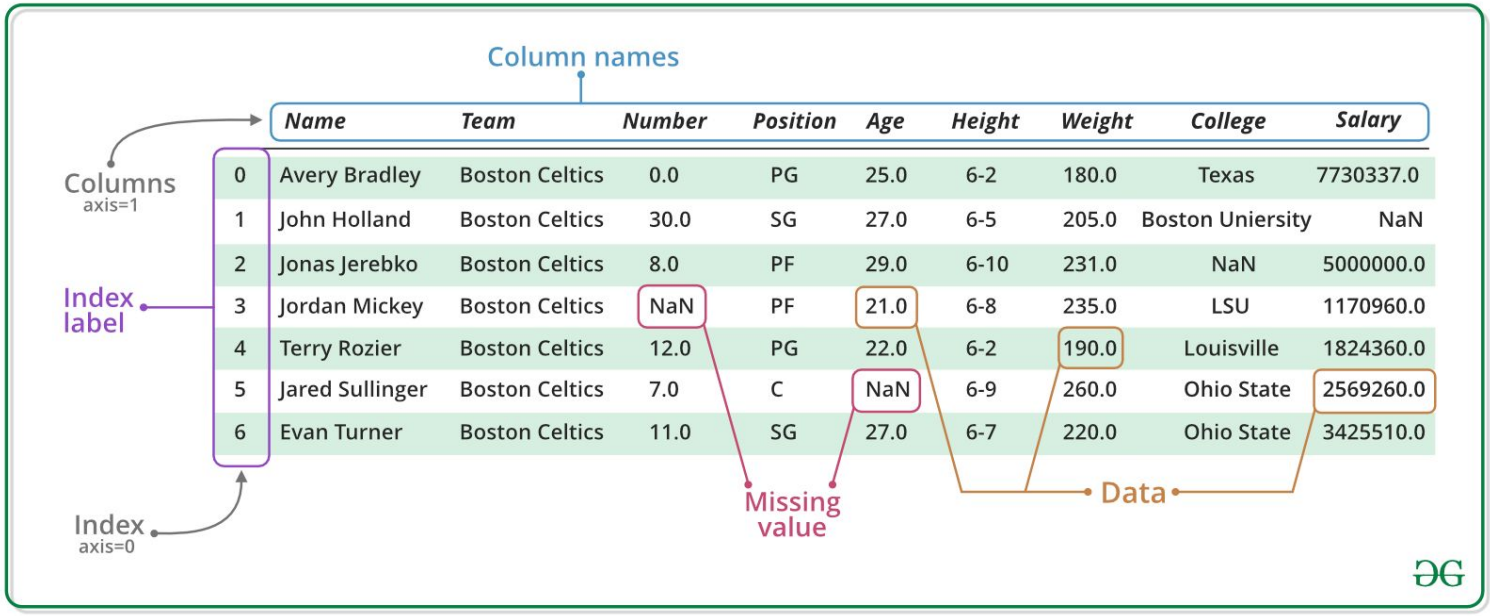
Pandas Data structure



Data Frame Basics



Data Frame Basics



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Data Frame Creation Syntax

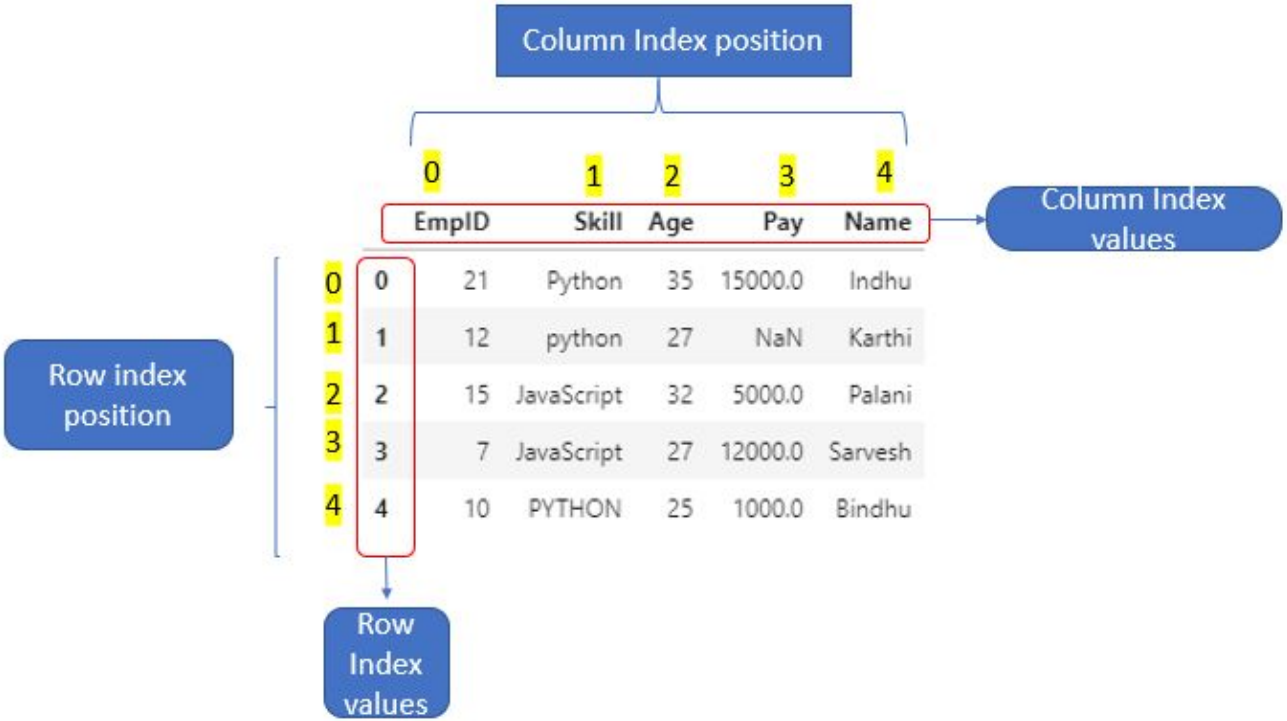
```
data = {'name': ['Alice', 'Bob', 'Charlie'],
        'age': [25, 30, 35],
        'city': ['New York', 'San Francisco', 'London']}
df = pd.DataFrame(data)
```

Data Indexing

Data indexing and selecting refers to the process of selecting subsets of data from a larger dataset in a specific manner. Indexing allows you to access specific rows and columns within a DataFrame, while selecting allows you to filter data based on certain criteria.

In pandas, data indexing and selecting can be done using various techniques, such as **integer indexing, label indexing, boolean indexing, and slicing**. By using these techniques, you can extract a specific subset of data from a DataFrame, which is useful when you need to analyze only a portion of your data or when you want to perform operations on a specific part of your dataset.

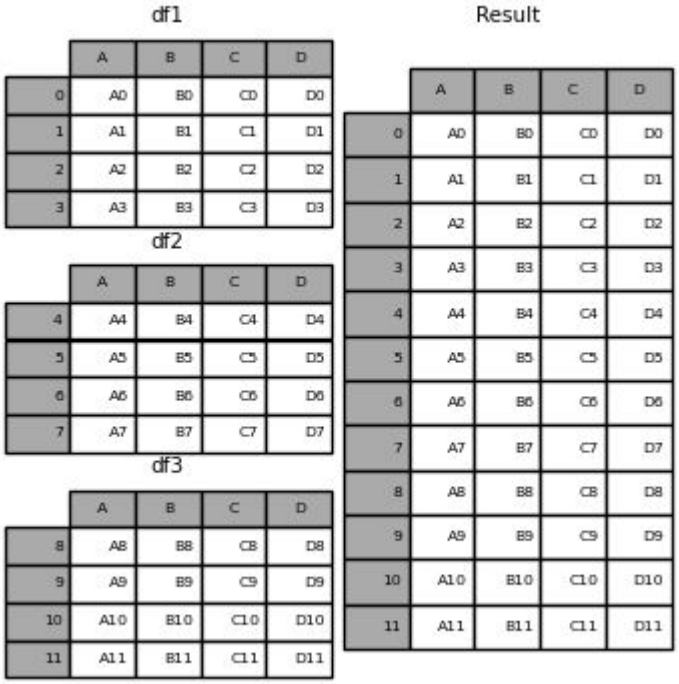
Data Indexing



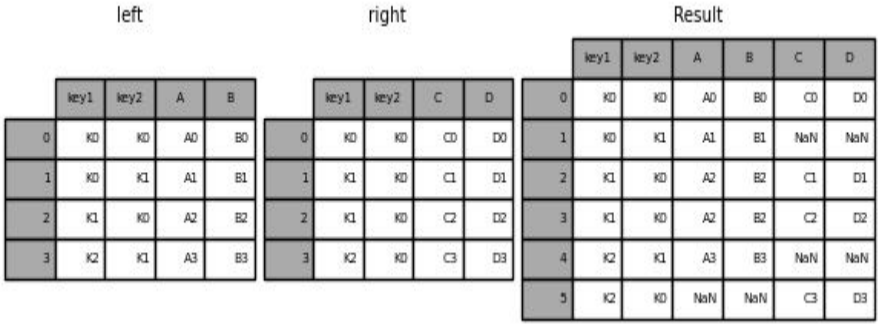
Data Aggregation (Groupby)

	Name	Team	Position	Age	Weight
0	Avery Bradly	Boston Celtics	PG	25.0	180.0
1	Jae Crowder	Boston Celtics	SF	25.0	235.0
2	John Holland	Boston Celtics	SG	27.0	205.0
3	R.j. Hunter	Boston Celtics	SG	22.0	185.0
4	Sergey Karasev	Brooklyn Nets	SG	22.0	208.0
5	sean Kilpatrick	Brooklyn Nets	SG	26.0	219.0
6	Shane Larkin	Brooklyn Nets	PG	23.0	175.0
7	Brook Lopez	Brooklyn Nets	C	28.0	275.0
8	Chris Johnson	Utah Jazz	SF	26.0	206.0
9	Trey Lyles	Utah Jazz	PF	20.0	234.0
10	Shelvin Mack	Utah Jazz	PG	26.0	203.0
11	Raul Pleiss	Utah Jazz	PG	24.0	179.0

Data Stacking (concat)



Vertically.



Data Reshaping.

Melt

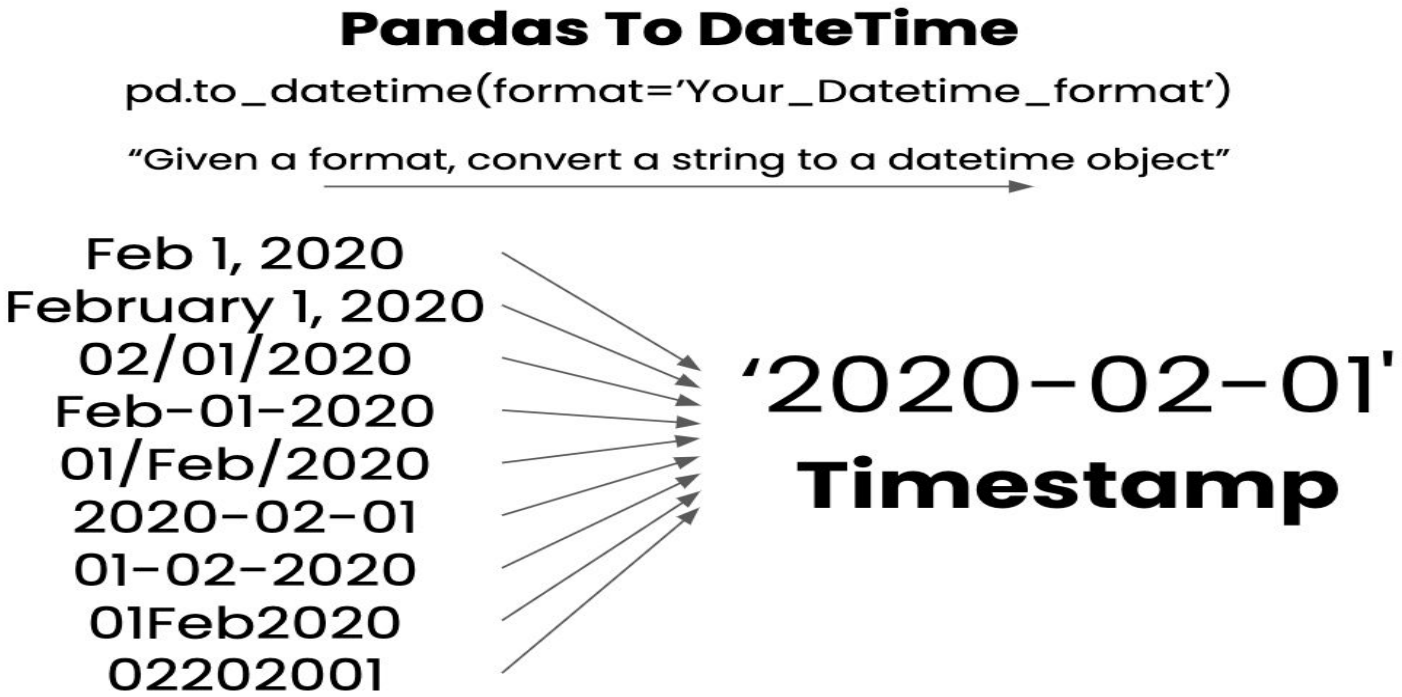
df3

	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150

```
df3.melt(id_vars=['first', 'last'])
```

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

Handling Dates and Times



Handling Dates and Times

```
In [1]: import pandas as pd
```

Are your dates split across multiple columns?

```
In [2]: df = pd.DataFrame([[12, 25, 2019, 'christmas'],
                           [11, 28, 2019, 'thanksgiving']],
                          columns=['month', 'day', 'year', 'holiday'])
        df
```

Out[2]:

	month	day	year	holiday
0	12	25	2019	christmas
1	11	28	2019	thanksgiving

Make a single datetime column with `to_datetime()`

```
In [3]: df['date'] = pd.to_datetime(df[['month', 'day', 'year']])
        df
```

Out[3]:

	month	day	year	holiday	date
0	12	25	2019	christmas	2019-12-25
1	11	28	2019	thanksgiving	2019-11-28

```
In [4]: df.dtypes
```

Out[4]:

```
month                int64
day                  int64
year                 int64
holiday             object
date        datetime64[ns]
dtype: object
```

Reading in/out Data from multiple sources.

```
# read data from CSV
df = pd.read_csv('data.csv')

# read data from Excel
df = pd.read_excel('data.xlsx')

# read data from SQL database
import sqlite3
conn = sqlite3.connect('database.db')
df = pd.read_sql('SELECT * FROM table', conn)
```

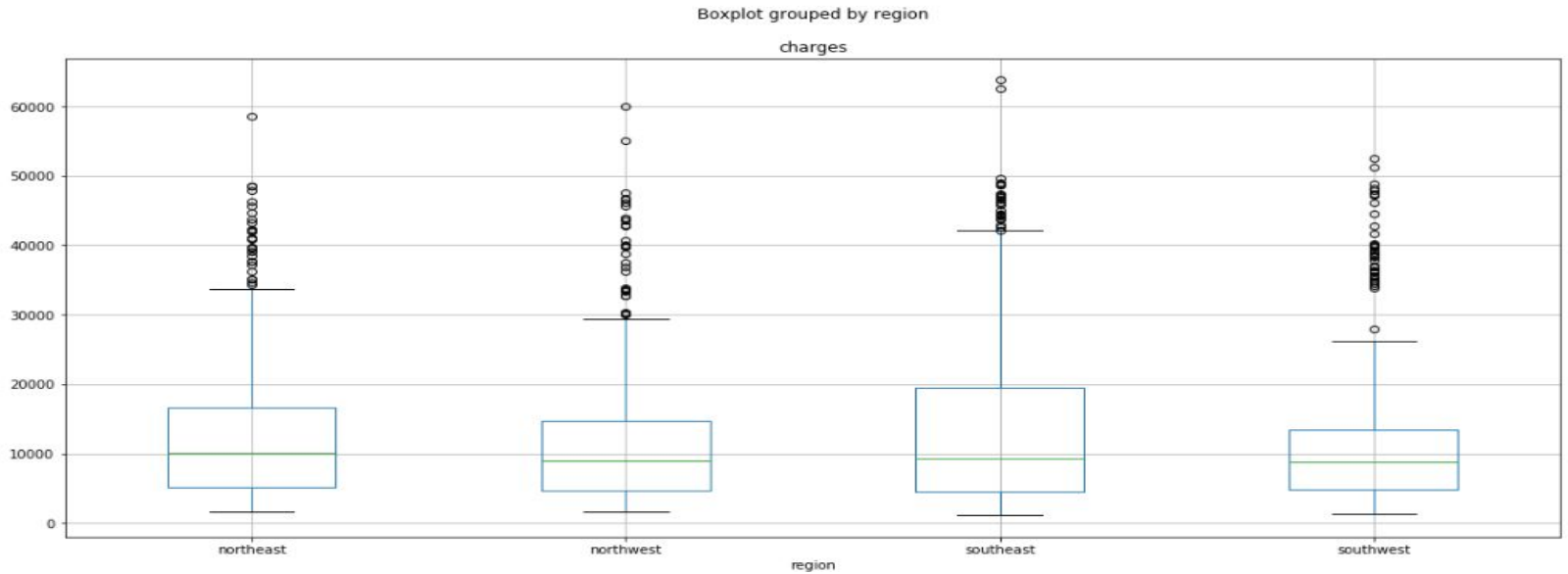
Handling Missing Data

```
# check for missing values
df.isna()

# fill missing values with a specific value
df.fillna(0)

# fill missing values with a method
df.fillna(method='ffill')

# drop rows with missing values
df.dropna()
```

Handling Simple Visualisation

```
df.boxplot(column="charges",by="region", figsize=(18, 8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8ea4eb70>
```



Questions?