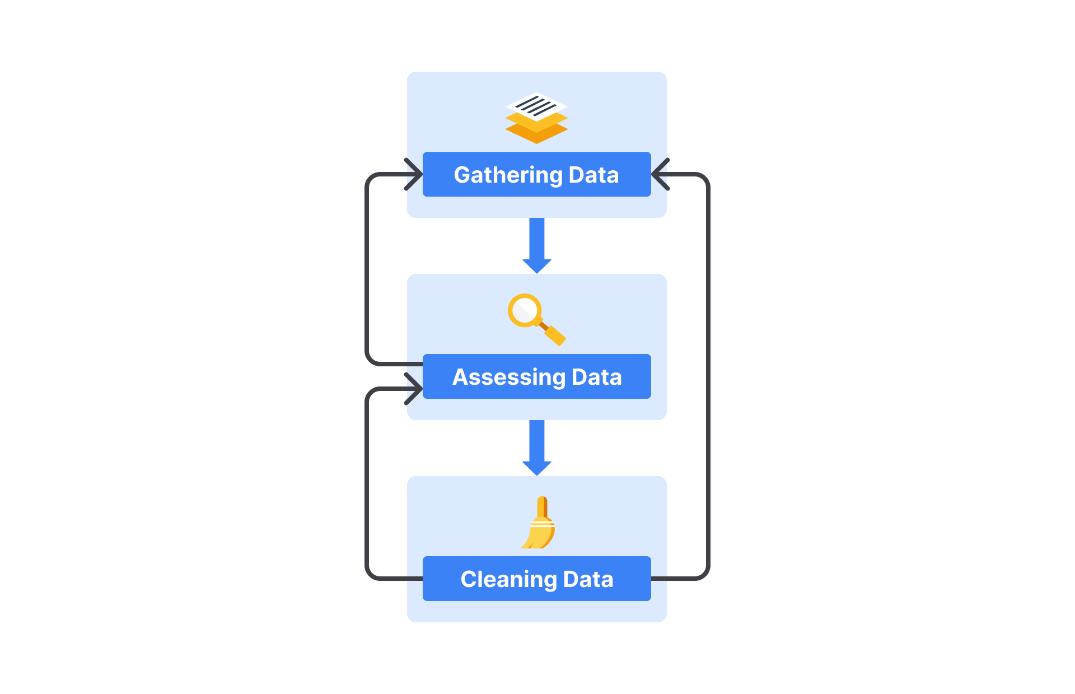
**Pengenalan Data Wrangling**

Selamat, Anda telah menempuh paruh pertama dalam perjalanan Anda untuk menyelesaikan kelas Belajar Analisis Data dengan Python. Sejauh ini, Anda telah mempelajari banyak hal mulai dari konsep dasar dalam analisis data, *descriptive statistics*, hingga berbagai pertimbangan dalam pengolahan data.

Nah, sekarang kita akan membahas salah satu tahap awal dalam siklus proyek analisis data yaitu *data wrangling*. Data wrangling merupakan sebuah proses atau kumpulan kegiatan yang meliputi pengumpulan data (Gathering data), penilaian data (Assessing data), serta pembersihan data (Cleaning data) sebelum data digunakan dalam proses analisis data [4].

* **Gathering data**  
  Tahap data wrangling, dimulai dengan proses pengumpulan data. Pada proses ini kita akan mengumpulkan semua data yang dibutuhkan untuk menjawab semua pertanyaan atau masalah bisnis yang ingin kita hadapi.
* **Assessing data**  
  Setelah semua data yang dibutuhkan terkumpul, proses selanjutnya ialah penilaian terhadap data tersebut. Proses ini dilakukan untuk menilai kualitas dan struktur dari sebuah data. Selain itu, proses ini juga bertujuan untuk mengidentifikasi berbagai masalah yang terdapat dalam data, seperti missing value, unstandard value, dll.
* **Cleaning data**  
  Apabila pada proses sebelumnya kita menemukan masalah (missing value, outlier, dll.) yang terdapat di dalam sebuah data, masalah tersebut harus dibersihkan sebelum masuk tahap analisis data. Terdapat beberapa teknik yang dapat kita gunakan untuk membersihkan data. Seluruh teknik tersebut akan kita pelajari pada beberapa materi ke depan.

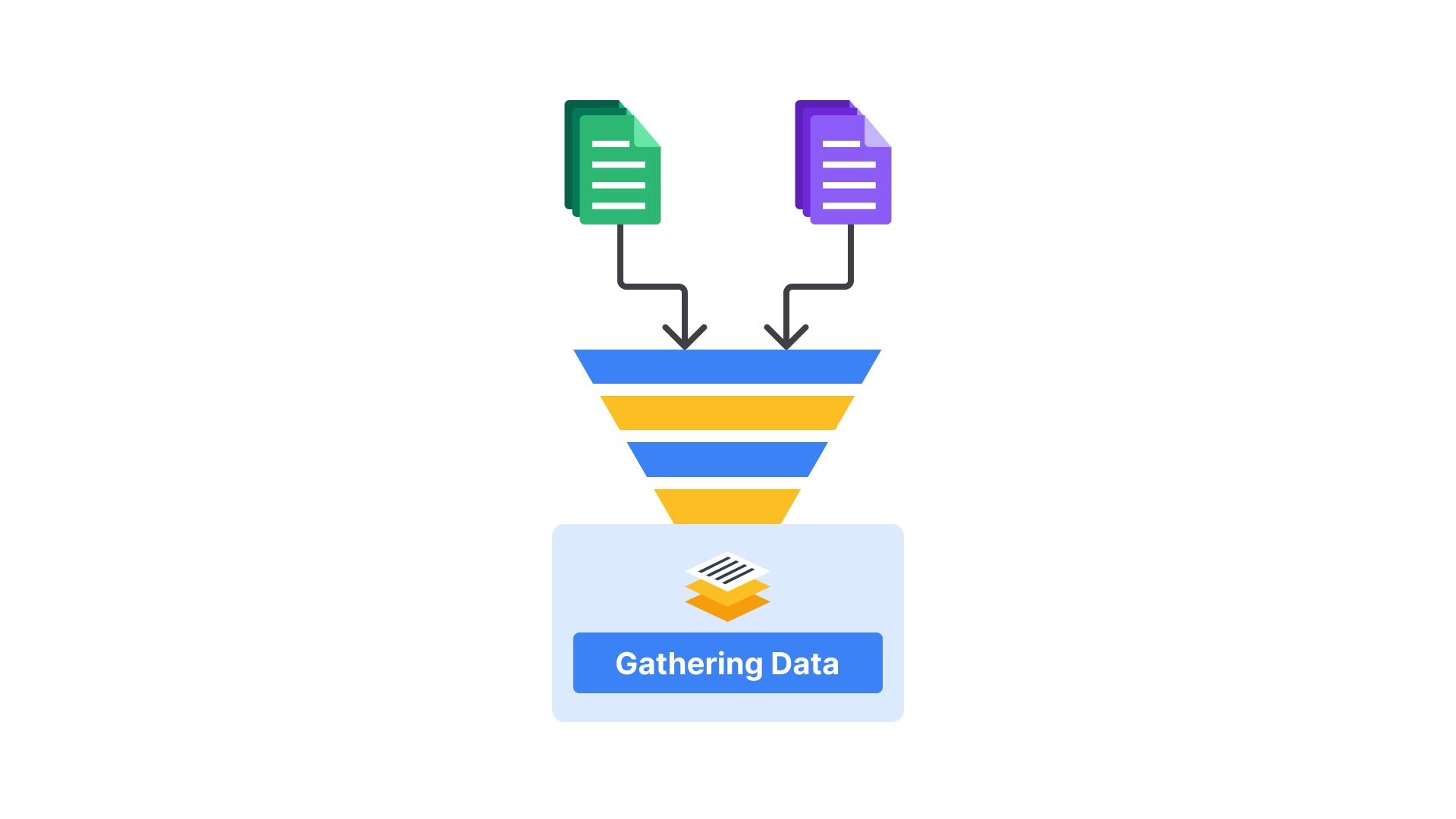


**Gathering Data**



Kalimat di atas merupakan salah satu ungkapan terkenal dari karakter detektif bernama Sherlock Holmes dalam novel karya Arthur Conan Doyle. Kalimat ini menunjukkan betapa pentingnya data sebagai *building block* dalam proses pembuatan kesimpulan dan pengambilan keputusan.

Sebagai calon praktisi data, tentunya kita juga sepakat dengan pendapat Sherlock Holmes tersebut. Bagi seorang praktisi data, data merupakan sebuah sumber daya paling penting untuk menyelesaikan berbagai permasalahan bisnis. Nah, pada materi kali ini kita akan membahas secara tuntas proses gathering data atau pengumpulan data.



Sebelum mulai, perlu Anda ketahui bahwa proses pengumpulan data merupakan salah satu tahapan paling menantang dalam proyek analisis data. Bergantung pada proyek analisis data yang dikerjakan, data yang Anda butuhkan mungkin terdapat dalam berbagai sumber dan memiliki format yang berbeda-beda.

Materi kali ini tidak hanya mengajarkan Anda cara membaca sebuah data, melainkan cara mengumpulkannya juga. Hal ini tentunya untuk menyiapkan Anda menjadi seorang praktisi data yang andal.

Nah, sebagai permulaan, mari kita kenalan dengan berbagai sumber data yang umum digunakan oleh para praktisi di industri.

**Berbagai Sumber Data**

Salah satu skill yang harus dimiliki oleh seorang praktisi data ialah mengetahui sumber data yang tepat untuk mengumpulkan data yang dibutuhkan. Oleh karena itu, sangat penting bagi kita untuk mengetahui berbagai sumber data yang umum digunakan.

Nah, berikut merupakan beberapa sumber data yang bersifat publik dan sering digunakan oleh para praktisi data.

* **Kaggle**  
  Jika stack overflow merupakan platform favorit bagi para *programmer*ketika menemukan eror, kaggle merupakan platform andalan bagi praktisi data untuk mencari dataset. Selain menyediakan dataset, kaggle juga menyediakan berbagai kompetisi dan *challenge* terkait data science dan machine learning. Kaggle juga memungkinkan para praktisi data dan machine learning saling berbagi kode dan pengetahuan terkait data serta machine learning. Untuk mulai menggunakan platform ini, Anda perlu membuat akun terlebih dahulu melalui tautan berikut: [kaggle](https://www.kaggle.com/" \t "_blank).
* **UCI Machine Learning Repository**  
  UCI Machine Learning Repository merupakan sebuah repositori yang menampung berbagai dataset yang bersifat publik. Seluruh dataset tersebut umumnya digunakan untuk kebutuhan akademik maupun bahan latihan bagi calon praktisi data dan machine learning pemula. Anda dapat mengakses repositori ini melalui tautan berikut: [UCI Machine Learning Repository](https://archive.ics.uci.edu/).
* **Google Dataset Search**  
  Google Dataset Search merupakan sebuah search engine yang disediakan oleh Google. Ia dibuat untuk mempermudah para praktisi data dan juga *researcher*dalam mencari dataset yang bersifat publik. Cara penggunaannya mirip seperti ketika kita menggunakan Google Search. Anda dapat mengaksesnya melalui tautan berikut: [Google Dataset Search](https://datasetsearch.research.google.com/).
* **Satu Data Indonesia**  
  Tahukah Anda bahwa pemerintah Indonesia sebetulnya telah menyediakan sebuah platform bernama Satu Data Indonesia atau sering disingkat SDI. Platform ini dibuat sebagai bentuk kebijakan tata kelola data pemerintah yang bertujuan untuk menciptakan data berkualitas dan mudah diakses. Platform ini dikelola oleh Sekretariat Satu Data Indonesia tingkat Pusat yang berada di naungan Kementerian Perencanaan Pembangunan Nasional/Bappenas. Untuk mengakses platform ini, silakan mengunjungi tautan berikut: [Satu Data Indonesia](https://data.go.id/home).

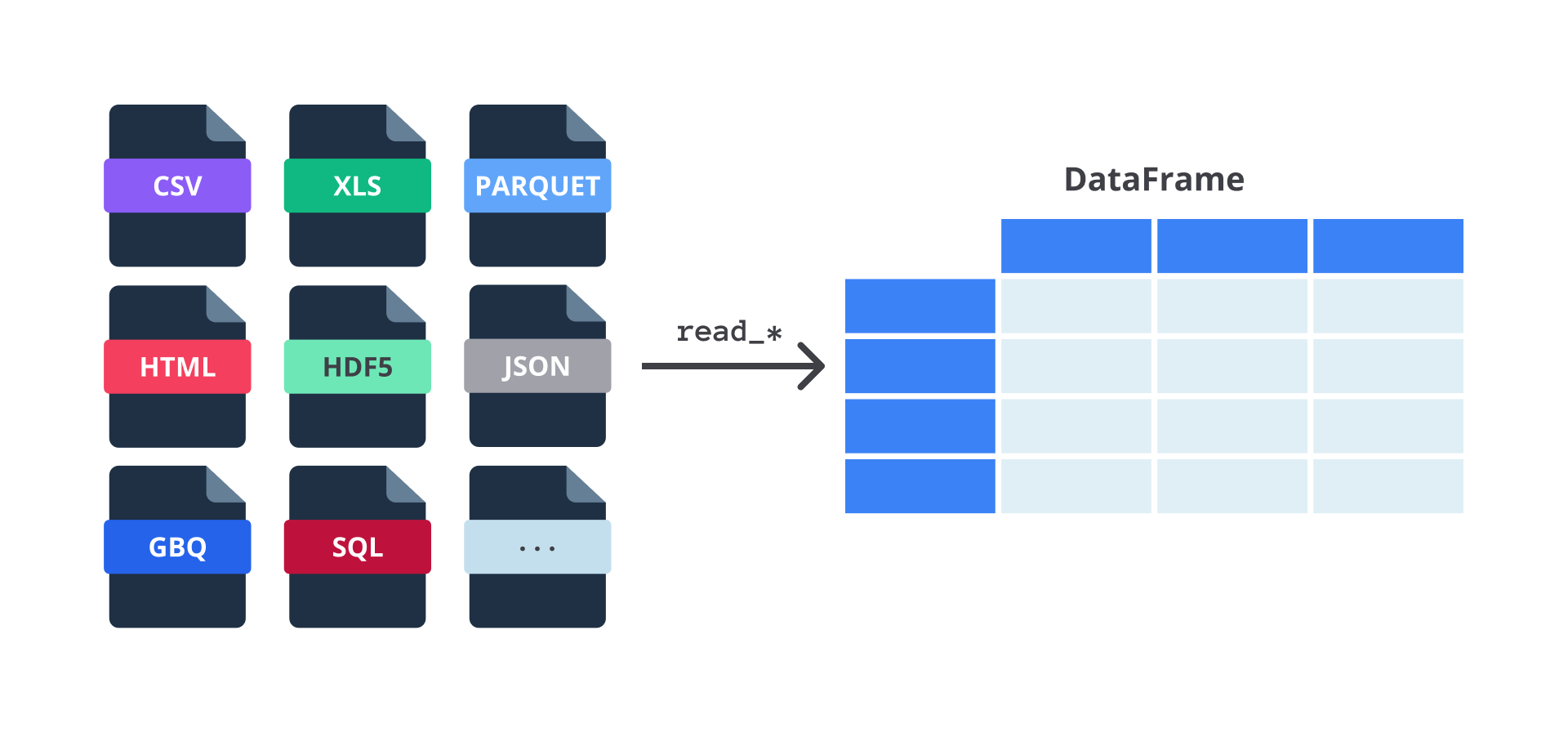
Selain menggunakan berbagai sumber dataset publik tersebut, para praktisi data juga seringkali menggunakan data internal yang bersifat privat. Data seperti ini umumnya disimpan dalam sebuah sistem pengolahan database yang hanya bisa diakses dan digunakan untuk kepentingan internal sebuah organisasi.

Nah, sekarang Anda telah mengenal beberapa sumber data yang umum digunakan oleh para praktisi data. Cara pengumpulan data melalui beberapa sumber data di atas cukup untuk menyelesaikan beberapa permasalahan umum yang Anda hadapi. Namun terkadang, pada beberapa kasus khusus, Anda mungkin perlu menggunakan teknik pengumpulan data lanjutan seperti web scraping yang di luar cakupan pembahasan kita pada kelas ini.

**Membaca Berbagai Tipe Data Menggunakan Pandas**

Pada pembahasan sebelumnya kita telah mengenal berbagai sumber data yang umum digunakan oleh praktisi data. Nah, sekarang kita mulai pembahasan tentang cara membaca data menggunakan library pandas.

Seperti yang telah kita bahas sebelumnya, pandas merupakan sebuah library Python yang spesifik digunakan untuk memanipulasi dan menganalisis data. Dalam mendukung hal tersebut, pandas menyediakan beberapa function yang dapat digunakan untuk membaca atau mengakses data menjadi sebuah DataFrame.



Saat ini, Pandas mendukung pembacaan atau pengaksesan data dari berbagai format berkas data dan sumber data seperti pada contoh berikut. Namun, perlu Anda ingat bahwa beberapa kode di bawah ini hanya digunakan sebagai contoh dan bukan untuk dijalankan.

* **Format berkas CSV**  
  Berkas CSV (*Comma Separated Values*) merupakan format berkas data tabel yang paling sering digunakan dan telah menjadi standar dalam industri. Ia menggunakan koma (,) sebagai pemisah (sering disebut sebagai *delimiter*) antar nilai dalam satu baris.  
    
  Pandas menyediakan sebuah function read\_csv()untuk membaca berkas CSV. Berikut merupakan contoh penggunaannya.

|  |
| --- |
| import pandas as pd    df = pd.read\_csv("data.csv", delimiter=",") |

* **Format berkas XLSX atau XLS**  
  Berkas XLSX atau XLS merupakan format berkas *spreadshee*t yang dibuat menggunakan aplikasi Microsoft Excel. Pandas juga menyediakan function read\_excel() untuk membaca berkas data dengan format XLSX atau XLS. Berikut merupakan contoh kodenya.

|  |
| --- |
| import pandas as pd    df = pd.read\_excel("data.xlsx", sheet\_name="Sheet1") |

* **Format berkas JSON**  
  Berkas JSON (*JavaScript Object Notation*) merupakan format berkas data lain yang paling sering digunakan di industri. Ia sering digunakan karena berukuran kecil, mudah dibaca dan ditulis oleh manusia, serta mudah diproses oleh mesin. JSON memiliki struktur data yang mirip seperti *data structure dictionary* dalam Python yang terdiri dari pasangan *keys* dan *values*.  
    
  Sebagai library pengolahan data yang andal, pandas juga menyediakan function read\_json()untuk membaca berkas data berformat JSON. Berikut merupakan contoh penggunaan function tersebut.

|  |
| --- |
| import pandas as pd    df = pd.read\_json("data.json") |

* **Format berkas HTML**  
  HTML atau dikenal juga sebagai *HyperText Markup Language* merupakan sebuah markup language standar yang digunakan untuk merancang tampilan sebuah dokumen/halaman di *web browser*. Untuk membaca berkas ini, pandas menyediakan function read\_html(). Ia akan menerima inputan berupa HTML string, HTML file, atau URL dan akan mengurai tabel HTML ke dalam bentuk list. List ini berisi kumpulan DataFrame yang diurai dari tabel HTML tersebut. Berikut merupakan contoh kode penggunaan function tersebut.

|  |
| --- |
| url = "https://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list"  df = pd.read\_html(url)[0] |

* **Format berkas XML**  
  Format data selanjutnya yang akan kita bahas ialah XML. Ia merupakan singkatan dari Extensible Markup Language yang sering digunakan untuk merepresentasikan berbagai struktur informasi, seperti dokumen, data, konfigurasi, dll.  
    
  Pandas menyediakan function read\_xml()untuk membaca format data ini. Berikut merupakan contoh kode penggunaannya.

|  |
| --- |
| import pandas as pd    df = pd.read\_xml("https://www.w3schools.com/xml/books.xml") |

* **Akses data dari SQL database**  
  Selain membaca data dari berbagai format, library pandas juga memungkinkan kita untuk mengakses data langsung dari sebuah database, seperti PostgreSQL, MySQL, dll. Tentunya untuk mengakses database tersebut kita membutuhkan library pendukung yaitu [SQLAlchemy](https://docs.sqlalchemy.org/en/20/dialects/index.html" \t "_blank).  
    
  Untuk berinteraksi dengan database, pandas menyediakan tiga function seperti berikut.
* read\_sql\_table()untuk membaca SQL database table dan mempresentasikannya ke dalam bentuk pandas DataFrame.

|  |
| --- |
| import pandas as pd  import sqlalchemy as sqla    db = sqla.create\_engine("sqlite:///mydata.sqlite")    pd.read\_sql\_table("table\_name", db) |

* read\_sql\_query()untuk membaca SQL query dan mempresentasikannya ke dalam bentuk pandas DataFrame.

|  |
| --- |
| import pandas as pd  import sqlalchemy as sqla    db = sqla.create\_engine("sqlite:///mydata.sqlite")    pd.read\_sql\_query("SELECT \* FROM table\_name", db) |

* read\_sql()untuk membaca SQL query atau table dan mempresentasikannya ke dalam bentuk pandas DataFrame.

|  |
| --- |
| import pandas as pd  import sqlalchemy as sqla    db = sqla.create\_engine("sqlite:///mydata.sqlite")    pd.read\_sql("SELECT \* FROM table\_name", db) |

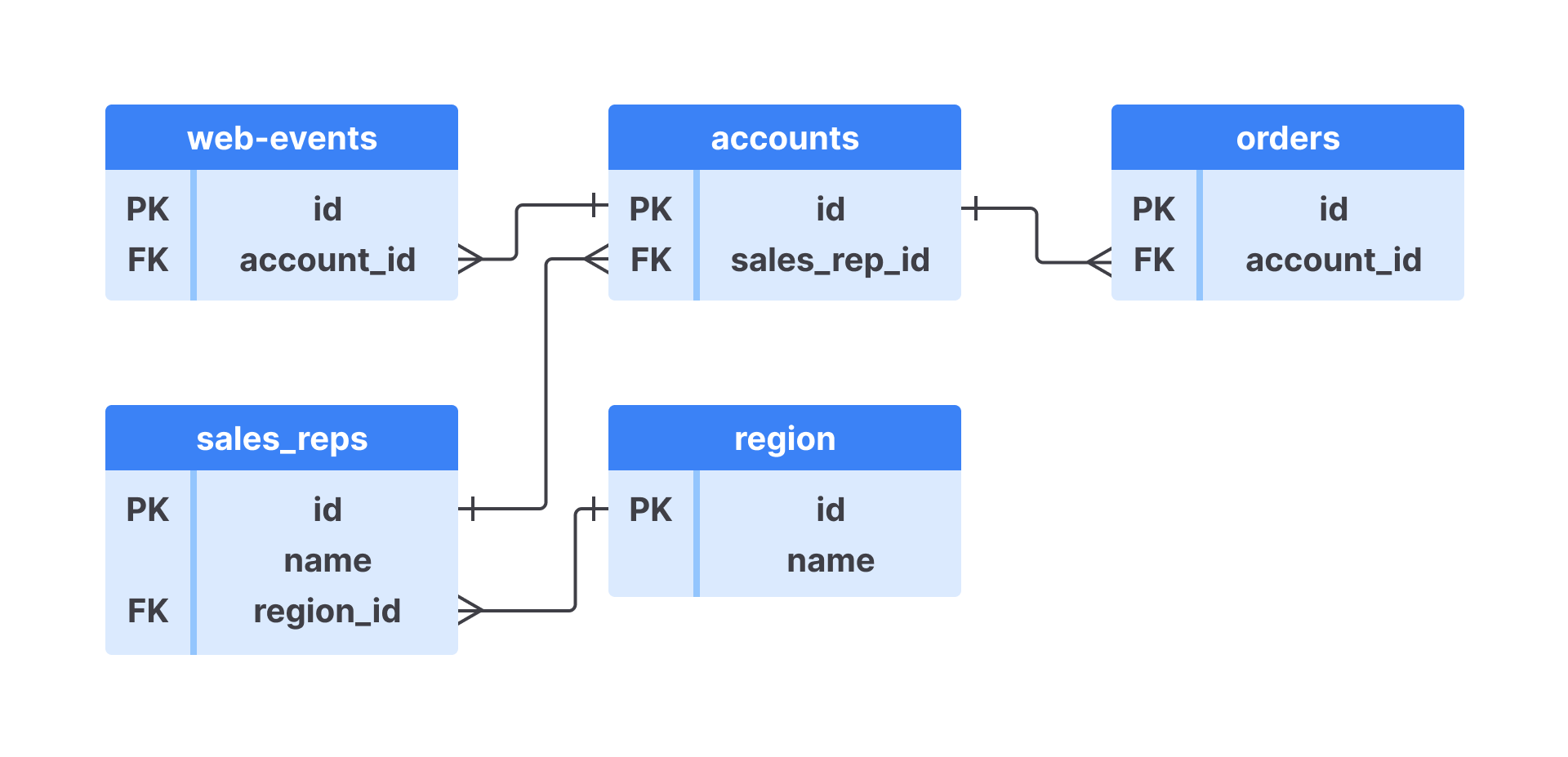
Nah, itulah beberapa contoh format data yang umum dijumpai di industri. Tentunya masih ada beberapa format data lain yang belum sempat kita bahas pada kesempatan ini. Jika Anda ingin melihat lebih detail terkait hal ini, silakan kunjungi tautan berikut: [pandas IO tool documentation](https://pandas.pydata.org/docs/user_guide/io.html" \t "_blank).

**Menggabungkan Beberapa Data Menjadi Satu DataFrame**

Seperti yang telah kita bahas sebelumnya, untuk menyelesaikan sebuah permasalahan bisnis, terkadang kita perlu menggunakan lebih dari satu tabel data. Tentunya untuk melakukan analisis kita perlu menggabungkan data tersebut.

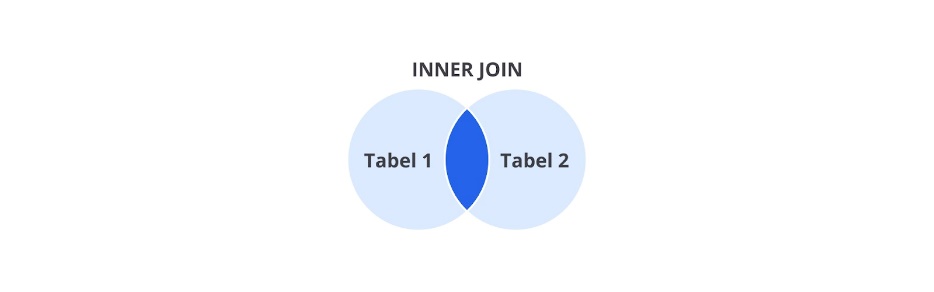
Salah satu teknik penggabungan data yang paling sering digunakan ialah *merge* atau *join*. Bagi Anda yang familier dengan *relational database* tentunya sudah tidak asing lagi dengan istilah tersebut. Ia merupakan teknik untuk menggabungkan dua tabel data menggunakan *primary key* (PK) dan *foreign key*(FK).

Primary key merupakan sebuah kolom dengan nilai unik yang merepresentasikan suatu data dalam sebuah tabel. Di lain sisi, foreign key merupakan kolom yang berisi primary key dari tabel lain. Ia digunakan untuk mereferensikan data dari tabel lain hingga terbentuk sebuah relationship antar tabel. Inilah yang menjadi kunci dalam relational database [9].

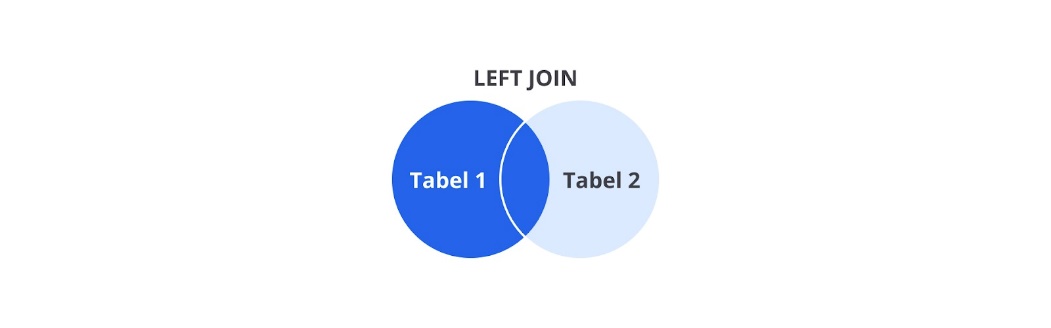


Berdasarkan cara penggabungannya, proses merge atau join dapat dibagi menjadi empat jenis yaitu seperti berikut.

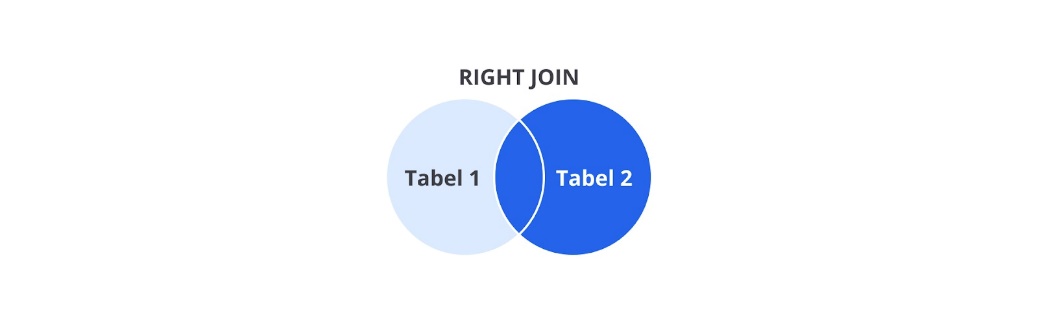
* **Inner**  
  Inner join merupakan proses join yang hanya mengambil nilai yang bersesuaian di kedua tabel.



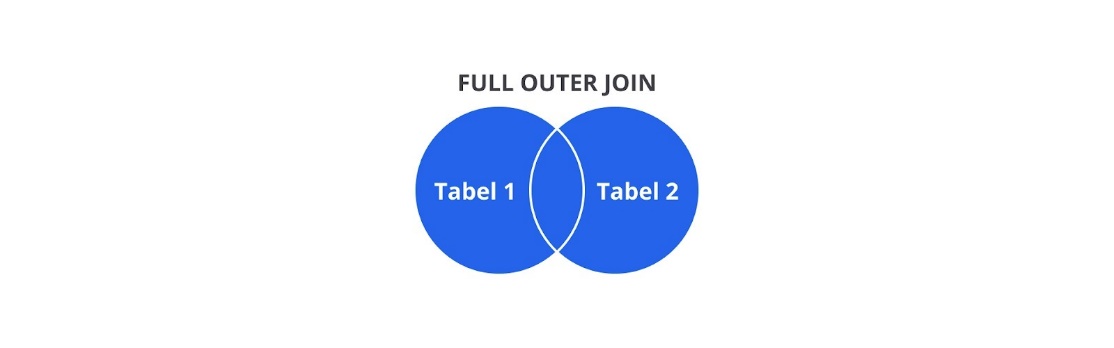
* **Left**  
  Left join merupakan proses join yang akan mengambil semua nilai dari tabel kiri beserta nilai yang bersesuaian dari tabel kanan.



* **Right**  
  Right join merupakan proses join yang akan mengambil semua nilai dari tabel kanan beserta nilai yang bersesuaian dari tabel kiri. Ia merupakan kebalikan dari left join.



* **Outer**  
  Outer join atau sering juga disebut *full outer join* merupakan proses join yang akan mengambil semua nilai dari kedua tabel. Ia merupakan gabungan dari left dan right join.



Sebagai tool andalan dalam pengolahan dan analisis data, pandas menyediakan sebuah function bernama merge(). Ia dapat digunakan untuk menggabungkan dua buah DataFrame. Berikut merupakan contoh kode penggunaannya.

|  |
| --- |
| import pandas as pd    product\_df = pd.read\_csv("product.csv")  orders\_df = pd.read\_csv("orders.csv")    new\_order\_df = pd.merge(  left=product\_df,  right=orders\_df,  how="inner",  left\_on="product\_id",  right\_on="product\_id"  ) |

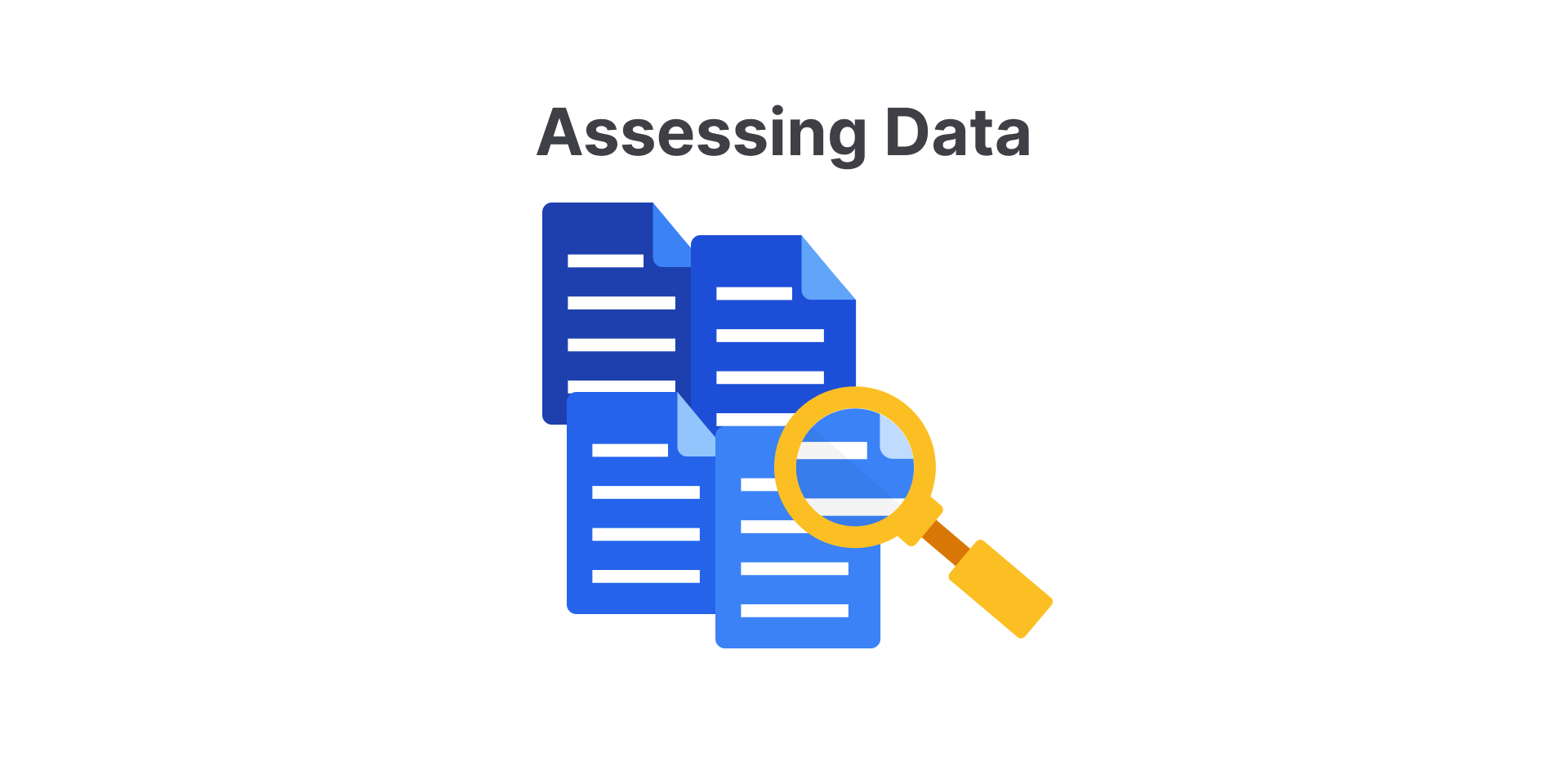
Kode di atas akan menghasilkan sebuah DataFrame baru yang hanya mengambil nilai yang bersesuaian dari kedua DataFrame (product\_df dan orders\_df) tersebut. Proses ini dilakukan dengan menyesuaikan nilai pada kolom product\_id yang berperan sebagai *primary key* dari product\_df dan *foreign key* dari orders\_df.

Oke, itulah berbagai konsep dan tahapan penting dalam proses gathering data. Penerapan dari materi ini dapat Anda simak dalam materi Latihan Data Wrangling.

**Assessing Data**

Sampai dengan tahap ini Anda telah memahami berbagai hal dalam proses g*athering data*. Data yang telah dikumpulkan tersebut harus diperiksa terlebih dahulu sebelum masuk ke tahap analisis.

Pemeriksaan data ini dilakukan dengan menjalankan proses assessing data. Ia merupakan proses yang bertujuan untuk mengidentifikasi masalah yang terdapat dalam data dan memastikan data tersebut berkualitas.

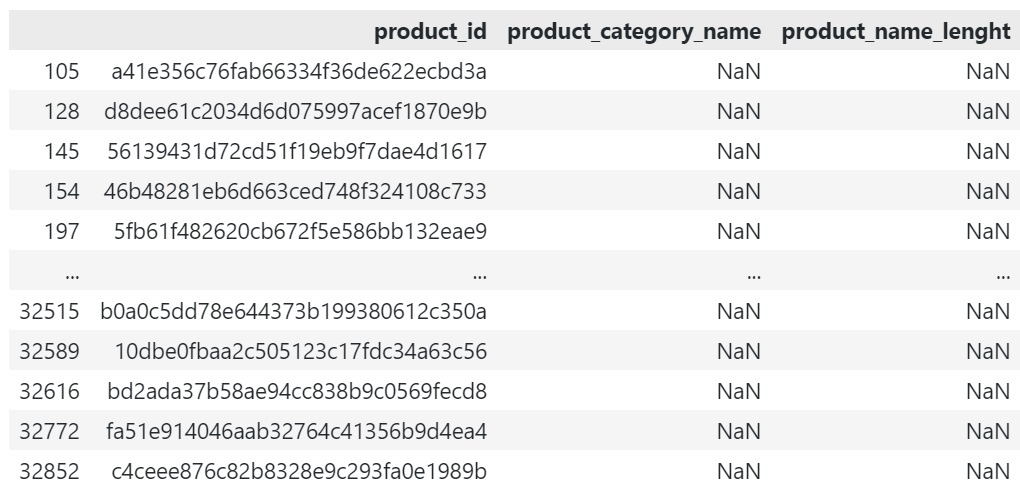


Mungkin beberapa dari Anda penasaran bagaimana cara mengidentifikasi masalah yang terdapat dalam sebuah data? Nah, materi kali ini dibuat khusus untuk menjawab pertanyaan tersebut. Sebagai permulaan, yuk, kita mengenal berbagai masalah yang umum dijumpai dalam sebuah data.

**Masalah Umum Dijumpai dalam Sebuah Data**

Seperti yang telah kita bahas sebelumnya, kebersihan dan kualitas dari sebuah data merupakan tantangan utama dalam proyek analisis data di industri. Data yang kotor umumnya memiliki masalah dalam kontennya. Berikut beberapa masalah yang umum dijumpai dalam sebuah data.

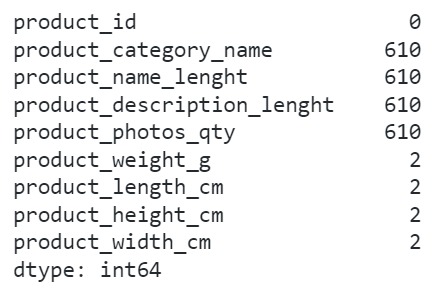
* **Missing value**  
  Missing value merupakan salah satu masalah yang paling sering dijumpai dalam proyek analisis data di industri. Masalah ini muncul karena adanya nilai yang hilang dari sebuah data dan biasanya direpresentasikan sebagai nilai **NaN** dalam library pandas. Hal ini biasanya terjadi karena adanya human error, masalah privasi, proses merging/join, dll.



Library pandas menyediakan sebuah method bernama isnull() atau isna() untuk mengidentifikasi missing value dalam sebuah DataFrame. Keduanya sering dipadukan dengan method sum()untuk menghitung total missing value pada setiap kolom dalam sebuah DataFrame. Berikut merupakan contoh penerapan kodenya.

|  |
| --- |
| import pandas as pd  product\_df = pd.read\_csv("product.csv")    product\_df.isnull().sum() |

Gambar di bawah ini merupakan contoh keluaran dari hasil pengecekan missing value. Pada contoh ini, kita menemukan cukup banyak missing value pada kolom **product\_category\_name**, **product\_name\_lenght**, dan **product\_description\_lenght**.



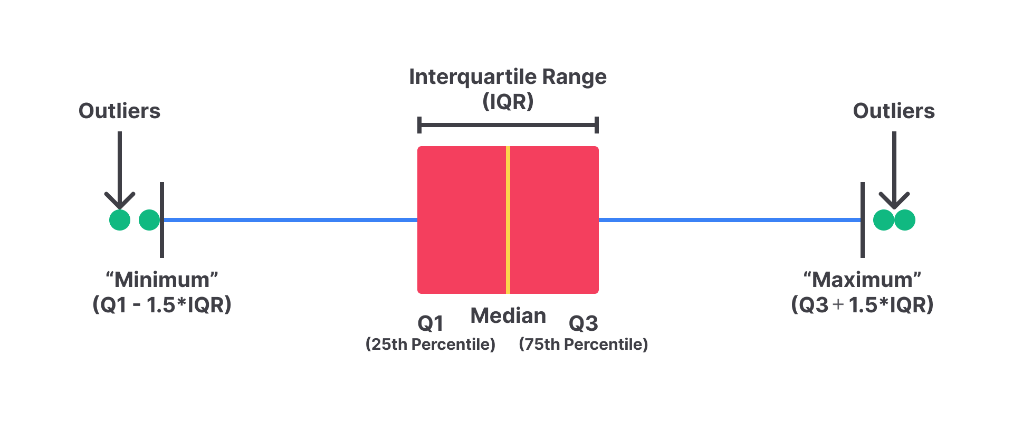
* **Invalid value**  
  Selain missing value, invalid value juga merupakan masalah lain yang sering dijumpai di industri. Masalah ini muncul ketika terdapat beberapa nilai yang tidak masuk akal, tidak sesuai dengan ketentuan, dan *background knowledge* dari data tersebut.  
    
  Sebagai contoh, data customer id haruslah bersifat unik dan memenuhi ketentuan tertentu seperti jumlah karakter serta komposisinya. Jika terdapat customer id yang tidak memenuhi ketentuan tersebut, akan dianggap sebagai invalid value.  
    
  Untuk mendeteksi masalah seperti ini, kita membutuhkan teknik sedikit *advance*. Salah satu teknik yang paling sering digunakan ialah teknik *filtering data* menggunakan *regex*.
* **Duplicate data**  
  Duplicate data merupakan masalah lain yang umum dijumpai di industri. Ia terjadi ketika terdapat sebuah observasi (semua nilai dalam satu unit baris) yang memiliki nilai yang sama persis pada setiap kolomnya.  
    
  Pandas menyediakan sebuah method duplicated()untuk mengidentifikasi apakah terdapat duplikasi pada sebuah DataFrame. Berikut merupakan contoh penerapannya.

|  |
| --- |
| import pandas as pd    url = "https://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list"  df = pd.read\_html(url)[0]  df.duplicated().sum() |

* **Inaccurate value**  
  Inaccurate value merupakan masalah yang muncul ketika nilai dalam sebuah data tidak sesuai dengan hasil observasi. Masalah ini umumnya muncul karena adanya human error atau sistem error.
* **Inconsistent value**  
  Inconsistent value adalah masalah yang muncul ketika sebuah data memiliki nilai yang tidak konsisten baik dari segi satuan maupun ketentuan penilaian. Inkonsistensi ini umumnya muncul karena adanya perbedaan standar dalam proses pengumpulan nilai.
* **Outlier**  
  Outlier atau dalam bahasa indonesia disebut pencilan merupakan titik data yang berada sangat jauh dari titik data yang lain dalam sebuah dataset. Nilai yang sangat jauh ini tentunya akan berdampak terhadap beberapa parameter statistik yang digunakan untuk menganalisis data, seperti nilai *mean* dan *standard deviation*.  
    
  Terdapat beberapa metode yang dapat digunakan untuk mengidentifikasi outlier dalam sebuah dataset. Metode yang paling sering digunakan ialah IQR method.  
    
  IQR method merupakan metode penentuan outlier berdasarkan nilai interquartile range (IQR). Ia mengidentifikasi outlier dengan cara membuat nilai *cut-off*sebagai faktor *k* (Umumnya kita menggunakan nilai 1.5 s/d 3) dari nilai IQR. Cut-off tersebut selanjutnya akan digunakan untuk menghitung nilai ambang batas (*boundary values*). Nilai tersebut dibagi menjadi dua yaitu ambang batas minimum dan maksimum. Semua titik data lebih kecil dari ambang batas minimum atau lebih besar dari ambang batas maksimum akan dianggap sebagai outlier.  
    
  Berikut merupakan contoh kode untuk mengidentifikasi outlier menggunakan IQR method.

|  |
| --- |
| import numpy as np    q25, q75 = np.percentile(data, 25), np.percentile(data, 75)  iqr = q75 - q25  cut\_off = iqr \* 1.5  minimum, maximum = q25 - cut\_off, q75 + cut\_off    outliers = [x for x in data if x < minimum or x > maximum] |

Metode lain yang bisa digunakan ialah box plot. Ia merupakan bentuk visual untuk merepresentasikan nilai IQR beserta ambang batas bawah dan atas dari sebuah data. Hal ini tentunya akan membantu kita mengidentifikasi outlier secara lebih mudah yaitu melalui bentuk visual.



*Yey*, sekarang Anda telah memahami berbagai masalah yang umum dijumpai dalam sebuah data di industri. Selain itu, Anda juga telah memahami berbagai teknik dan metode untuk mengidentifikasi masalah tersebut.

Jika terdapat sebuah masalah, tentunya kita harus menyelesaikan masalah tersebut. Nah, pada materi berikutnya, kita akan membahas berbagai teknik untuk membersihkan dan menyelesaikan masalah yang terdapat dalam data. *So, are you ready guys*?

**Cleaning Data**

Pada materi sebelumnya, kita telah membahas berbagai masalah yang umum dijumpai dalam sebuah data di industri beserta cara mengidentifikasinya. Sekarang kita akan membahas berbagai teknik untuk menyelesaikan masalah tersebut. Hal Ini menandakan kita telah memasuki proses terakhir dalam data wrangling yaitu cleaning data atau pembersihan data.

Sama halnya dengan pakaian kotor yang harus dibersihkan, data yang kotor juga harus kita bersihkan untuk memastikan ia tidak mempengaruhi hasil analisis yang akan kita lakukan nantinya. Nah, pada kesempatan ini, kita akan mengupas tuntas berbagai teknik dan konsep penting dalam proses pembersihan data.

Secara umum, proses pembersihan data dapat dibagi ke dalam tiga tahapan, yaitu *define*, *code*, dan *test*[4].

* **Define:** pada tahap ini, kita akan membuat rancangan tahapan serta metode pembersihan data berdasarkan masalah yang ditemukan dalam proses assessing data. Hal ini dapat dijadikan sebagai dokumentasi untuk memastikan orang lain memahami setiap tahapan dalam pembersihan data yang akan kita lakukan.
* **Code:** setelah membuat rancangan pembersihan data, tahap selanjutnya ialah mengonversi hal tersebut menjadi sebuah kode program yang dapat dijalankan.
* **Test:** setelah menjalankan kode program untuk membersihkan data, kita perlu memeriksa kembali data yang telah dibersihkan tersebut. Hal ini untuk memastikan proses pembersihan data dilakukan sesuai ekspektasi kita.

Ketiga tahapan tersebut telah menjadi standar best practice dalam proses pembersihan data. Nah, sebelum mengaplikasikan ketiga tahapan tersebut, Anda perlu memahami berbagai metode untuk membersihkan data terlebih dahulu.

**Teknik untuk Mengatasi Missing Value**

Seperti yang telah kita bahas pada materi sebelumnya, missing value merupakan salah satu masalah yang paling sering dijumpai di industri. Nah, pada kesempatan ini, kita akan berkenalan dengan berbagai metode yang umum digunakan untuk mengatasi missing value.

Secara garis besar, terdapat tiga metode dalam mengatasi missing value antara lain seperti berikut.

* **Dropping**  
  Metode pertama dan yang paling mudah dalam mengatasi missing value adalah dropping. Pada metode ini, kita akan menghapus seluruh baris atau kolom yang memiliki missing value. Untuk melakukannya, kita bisa menggunakan salah satu method dropna()yang disediakan oleh library pandas. Berikut merupakan contoh penerapan kodenya.

|  |
| --- |
| import pandas as pd    products\_df = pd.read\_csv("product.csv")    products\_df.dropna(axis=0, inplace=True) |

Pada contoh kode di atas, parameter axis=0 (menerima nilai 0 atau 1) menandakan kita ingin men-*drop* seluruh baris yang mengandung missing value. inplace=True menandakan kita ingin langsung menerapkan operasi tersebut ke dalam DataFrame products\_df. Anda dapat membaca dokumentasi berikut untuk lebih memahami penggunaan method ini: [pandas.DataFrame.dropna](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html" \t "_blank).  
  
Metode dropping ini memang terkesan mudah, tetapi perlu Anda ketahui bahwa ada banyak sekali pertimbangan yang perlu diperhatikan sebelum menggunakan metode ini. Salah satu pertimbangan penting ialah penerapan metode ini dapat menyebabkan kita kehilangan banyak informasi. Bergantung pada kasus dan jumlah data yang sedang ditangani, terkadang kita bisa mengabaikan hilangnya informasi ini. Sebagai contoh, jika bekerja dengan dataset yang sangat besar (puluhan ribu), kita bisa mengabaikan dampak tersebut. Sebaliknya, ketika bekerja menangani data berbentuk *time series* (data yang disusun berdasarkan urutan waktu) atau ukuran data yang kecil, kita perlu mempertimbangkan lagi penggunaan metode ini.

* **Imputation**  
  Metode lain yang umum digunakan untuk mengatasi missing value ialah metode imputation. Metode ini bekerja dengan cara mengisi (*fill*) missing value dengan nilai tertentu. Hal ini tentunya akan mencegah hilangnya informasi akibat missing value.  
    
  Pada data kontinu, kita bisa menggunakan nilai mean, median, atau mode sebagai pengganti missing value. Jika bekerja menggunakan data kategoris, kita dapat mengisi missing value dengan kategori yang paling sering muncul. Namun, perlu diingat bahwa pemilihan nilai pengganti ini harus didukung oleh *background knowledge*dari data tersebut. Pada beberapa kasus ada suatu nilai tertentu yang digunakan untuk mengganti missing value.  
    
  Sebagai library andalan kita, pandas telah menyediakan sebuah method bernama fillna() untuk mengganti missing value dengan nilai tertentu (Dokumentasi: [pandas.DataFrame.fillna](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html" \t "_blank)). Berikut merupakan contoh kode untuk mengganti missing value pada kolom agedengan nilai mean dari kolom tersebut.

|  |
| --- |
| import pandas as pd    data=pd.read\_csv('employee\_data.csv')    data.age.fillna(value=data.age.mean(), inplace=True) |

Nah, jika Anda perhatikan metode ini masih memiliki banyak kekurangan salah satunya ialah dapat mempengaruhi variance atau sebaran dari sebuah data. Selain itu, metode ini juga masih belum cukup baik untuk diterapkan pada data time series.

* **Interpolation**  
  Metode penanganan missing value terakhir yang akan kita bahas ialah interpolation (interpolasi). Sederhananya, interpolasi merupakan salah satu pendekatan numerik yang digunakan untuk menghitung titik data baru berdasarkan range data yang sudah ada. Perhitungan ini menggunakan sebuah persamaan garis *linear* ataupun *polynomial*. Perhitungan tersebut membuat metode ini sangat cocok digunakan untuk menangani missing value pada data time series.  
    
  Library pandas juga menyediakan method interpolate() yang bisa kita gunakan untuk menerapkan metode interpolasi dalam mengatasi missing value. Ketika menggunakan method ini, kita perlu mendefinisikan metode interpolasi yang ingin digunakan, seperti linear, polynomial, dll. Selain itu, kita juga perlu mendefinisikan parameter limit\_direction (*forward*, *backward*, dan*both*) untuk menspesifikkan arah konstruktif dari proses interpolasi. Berikut merupakan contoh kode untuk menggunakan method interpolate().

|  |
| --- |
| import pandas as pd    data=pd.read\_csv('bbca\_index.csv')    data.close\_price.interpolate(method='linear', limit\_direction='forward', inplace=True) |

Nah, untuk memahami lebih detail terkait method ini, Anda dapat membaca dokumentasi berikut: [pandas.DataFrame.interpolate](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html?highlight=interpolate" \l "pandas.DataFrame.interpolate" \t "_blank).

**Teknik untuk Mengatasi Outlier**

Oke, sebelumnya kita telah membahas berbagai metode untuk mengatasi missing value, Sekarang kita akan akan mengupas tuntas berbagai metode yang umum digunakan untuk mengatasi outlier.

Pada dasarnya, terdapat dua metode yang umum digunakan untuk mengatasi outlier yaitu drop dan imputation.

* **Drop**  
  Metode pertama yang paling mudah ialah men-*drop* atau menghapus seluruh baris yang mengandung outlier. Metode ini mampu mencegah outlier mempengaruhi hasil analisis yang kita buat.  
    
  Untuk menerapkan metode ini, kita bisa menggunakan method drop() yang disediakan oleh library pandas. Berikut merupakan contoh kode yang dapat Anda gunakan untuk melakukannya.

|  |
| --- |
| import pandas as pd    df = pd.read\_csv("data.csv")    Q1 = (df['TotalCharges']).quantile(0.25)  Q3 = (df['TotalCharges']).quantile(0.75)  IQR = Q3 - Q1    maximum = Q3 + (1.5\*IQR)  minimum = Q1 - (1.5\*IQR)    kondisi\_lower\_than = df['TotalCharges'] < minimum  kondisi\_more\_than = df['TotalCharges'] > maximum    df.drop(df[kondisi\_lower\_than].index, inplace=True)  df.drop(df[kondisi\_more\_than].index, inplace=True) |

* **Imputation**  
  Metode lain yang bisa Anda gunakan untuk menangani outlier ialah imputation. Konsepnya mirip seperti sebelumnya yaitu mengganti outlier dengan nilai tertentu. Nilai yang bisa kita gunakan ialah mean, median, dan mode. Selain itu, tidak jarang juga kita mengganti outlier dengan *boundary value*.  
    
  Untuk menerapkan metode ini, kita bisa menggunakan method mask() yang disediakan oleh library pandas. Method tersebut menerima parameter cond sebagai kondisi untuk memfilter nilai outlier. Berikut merupakan contoh kode untuk melakukannya.

|  |
| --- |
| df = pd.read\_csv("data.csv")    Q1 = (df['TotalCharges']).quantile(0.25)  Q3 = (df['TotalCharges']).quantile(0.75)  IQR = Q3 - Q1    maximum = Q3 + (1.5\*IQR)  minimum = Q1 - (1.5\*IQR)    kondisi\_lower\_than = df['TotalCharges'] < minimum  kondisi\_more\_than = df['TotalCharges'] > maximum    df.mask(cond=kondisi\_more\_than, maximum, axis=1, inplace=True)  df.mask(cond=kondisi\_lower\_than, minimum, axis=1, inplace=True) |

**Teknik untuk Mengatasi Duplicate Data**

Selain missing value dan outlier, duplicate data juga merupakan salah satu masalah yang umum dijumpai di industri. Pada materi sebelumnya, kita telah membahas cara untuk mendeteksinya. Nah, sekarang kita akan membahas cara untuk mengatasinya.

Ketika menemukan duplikasi pada data, tentunya kita harus menghilangkan atau menghapus duplikasi tersebut. Jika jumlah duplikasi dan ukuran data masih sedikit, mungkin kita bisa menghapusnya secara manual. Namun, solusi ini akan sangat merepotkan ketika jumlah duplikasi dan ukuran data membesar. Oleh karena itu, solusi seperti ini tidaklah *scalable*.

Untungnya, library pandas telah menyediakan sebuah method drop\_duplicates() untuk menghilangkan duplikasi dalam sebuah DataFrame. Berikut merupakan contoh kode untuk menggunakan method tersebut.

|  |
| --- |
| import pandas as pd    df = pd.read\_csv("data.csv")  df.drop\_duplicates(inplace=True) |