PENERAPAN FUNGSI MAP, FILTER DAN REDUCE UNTUK MENGOLAH LAPORAN POLISI DETROIT

Febiya Jomy Pratiwi, Alvia Asrinda Br.Ginting
Farrel Julio Akbar,Akmal Faiz Abdillah
Daffa Ahmad Naufal

Program Studi Sains Data Institut Teknologi Sumatera
Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan
Lampung 35365

Email:
febiya.122450074@student.itera.ac.id, alvia.122450077@student.itera.ac.id
farrel.122450110@student.itera.ac.id, akmal.122450114@student.itera.ac.id
daffa.122450137@student.itera.ac.id

I. Pendahuluan

1.1 Latar Belakang

Kami menggunakan data dari laporan Polisi Detroit yang dibaca dengan modul ‘csv’ dan diubah menjadi daftar dictionary. Data disaring menggunakan fungsi lambda dengan ‘filter()’ untuk menghapus baris yang datanya hilang di kolom Zip atau Neighborhood. Kemudian, kami menghitung rata-rata total waktu respons, waktu pengiriman, dan total waktu untuk Polisi Detroit menggunakan fungsi lambda dan ‘reduce()’ dari modul ‘functools’.

Untuk analisis yang lebih mendalam, data dibagi berdasarkan neighborhood menggunakan lambda dengan ‘map()’ atau ‘filter()’. Setelah itu, kami menghitung rata-rata total waktu respons, waktu pengiriman, dan total waktu untuk setiap neighborhood dengan fungsi lambda dan ‘reduce()’. Hasilnya disimpan dalam daftar kamus yang mencakup informasi waktu dan data populasi setiap neighborhood, serta total populasi Detroit.

1.2 Tujuan

Adapun tujuan dari laporan ini adalah sebagai berikut, yaitu:

1. Menggunakan fungsi lambda dengan ‘filter()’ untuk memastikan hanya data lengkap yang dianalisis.
2. Menggunakan fungsi lambda dan ‘reduce()’ untuk menghitung rata-rata waktu respons dan pengiriman.

3. Membagi dan menganalisis data berdasarkan neighborhood untuk membantu perencanaan dan pengambilan keputusan terkait keamanan kota.

II. Metode

2.1 pandas read_csv

Fungsi read_csv dalam Python digunakan untuk membaca file CSV (Comma Separated Values). CSV adalah singkatan dari nilai yang dipisahkan koma, yang berarti koma adalah delimiter default untuk file-file ini. CSV terdiri dari kolom dan baris, dan nilai sel disusun dalam format tabular. Alasan keberadaan file CSV termasuk data yang dikumpulkan secara manual, data yang diekstrak dan diunduh dari database, unduhan langsung dari alat atau situs web, web scraping, dan hasil dari menjalankan algoritma ilmu data [1].

2.2 dropna()

Fungsi dropna() digunakan untuk menangani nilai yang hilang dengan cara menghapusnya dari dataset kita. Kita telah melihat bahwa kita dapat menggunakan fungsi isnull() dan notnull() dari pustaka pandas untuk menentukan nilai null. Perlu diingat bahwa metode dropna() hanya mengembalikan salinan dari dataframe dengan menghapus baris yang berisi NaN. Dataframe asli tidak berubah. Jika dropna() diterapkan pada seluruh dataframe, maka akan menghapus semua baris dari dataframe, karena setidaknya terdapat satu nilai NaN dalam dataframe kita [2].

2.3 to_dict()

Fungsi to_dict() berperan dalam mengubah OrderedDict() yang terkompresi menjadi objek yang terdiri dari objek bersarang, mirip dengan dictionary biasa. Fungsi ini penting dalam analisis data karena memungkinkan pengguna mengonversi data dari DataFrame ke bentuk yang lebih mudah digunakan dalam analisis, seperti dictionary atau list [3].

2.4 filter()

Fungsi Filter() digunakan untuk menyaring data berdasarkan inisial nama kolom. Fungsi ini memungkinkan pengguna untuk mengembalikan rentang data yang memenuhi tinggi atau lebar yang sama seperti larik sumber. Fungsi filter() dalam Python mengambil fungsi dan daftar sebagai argumen, memberikan cara yang elegan untuk menyaring elemen dari urutan "order" yang mengembalikan nilai true [4].

2.5 reduce ()

Fungsi reduce() dalam Python digunakan untuk menghitung nilai berurutan dari sekumpulan nilai individual. Sebagai contoh, jika terdapat kumpulan angka, kita dapat menggunakan fungsi reduce() untuk menghitung jumlah dari angka-angka tersebut. Fungsi ini akan melakukan iterasi angka-angka secara berurutan, menghitung jumlah dari dua nilai pertama, menambahkan nilai ketiga ke jumlah tersebut, dan seterusnya, hingga setiap angka dalam kumpulan tersebut dimasukkan ke dalam penghitungan [5].

2.6 map ()

Dalam Python, terdapat sebuah fungsi bawaan yang disebut map() yang dapat digunakan untuk menerapkan sebuah fungsi pada setiap elemen dari objek yang diberikan menggunakan lambda. Fungsi map() akan mengambil item berikutnya dari setiap iterable dan menerapkan kedua nilai tersebut sebagai argumen ke fungsi yang diberikan. Dalam kasus ini, fungsi yang diberikan adalah lambda yang menghasilkan tiga tupel yang diinginkan, yaitu awal, akhir, dan jarak. Kami telah mendefinisikan lambda yang akan diterapkan ke setiap bagian perjalanan menggunakan fungsi map(). Perbedaan utama antara fungsi map() dan ekspresi generator adalah fungsi map() dapat menggunakan lambda atau definisi fungsi yang dapat digunakan kembali[6].

2.7 set()

Set dalam Python adalah koleksi data dengan elemen-elemen yang berbeda. Ada dua jenis set, yaitu set dinamis yang dapat diubah ukurannya dan set statis yang tidak dapat diubah isinya. Operasi-operasi seperti keanggotaan, gabungan, dan lainnya dapat dilakukan pada set[7].

2.8 list()

Fungsi list() digunakan untuk membuat daftar yang berisi berbagai tipe data seperti string, integer, float, boolean, dll. Dengan list(), Anda dapat mengelola dan mengakses data-data tersebut dalam satu struktur data terorganisir. Misalnya, Anda dapat membuat daftar yang berisi nama, umur, dan status pekerjaan seseorang dalam satu list. Ini memudahkan akses dan manipulasi data sesuai kebutuhan Anda [8].

2.9 sum()

Dalam membuat nilai baru, kita dapat menggunakan fungsi sum() yang sudah ada atau built-in function. Fungsi ini berguna untuk menjumlahkan nilai keluaran. Fungsi sum() menerima daftar tipe int atau float sebagai masukan dan mengembalikan jumlahnya [9].

2.10 len()

Fungsi len() memerlukan satu parameter utama yang berupa urutan atau koleksi, yang dapat berisi bytes, string, list, tuple, range, dictionary, set, dan frozenset. Fungsi len() adalah fungsi yang tersedia di Python yang berguna untuk menghitung jumlah karakter dalam parameter utama [10].

2.11 json.dump()

Json.dump() berfungsi untuk mengubah data ke dalam format JSON yang terenkripsi. Dengan melewatkan objek kamus ke dalam json.dump(), kita dapat menghasilkan string yang sesuai dengan format JSON yang dapat dibaca oleh aplikasi yang menggunakan JavaScript Object Notation (JSON) [11].

2.12 Lambda Function

Lambda adalah layanan komputasi yang memungkinkan Anda menjalankan kode tanpa perlu menyediakan atau mengelola server. Lambda biasanya digunakan untuk fungsi lain untuk mempercepat proses eksekusi tanpa membuat nama fungsi baru, seperti map(), filter(), dan sebagainya [12].

2.13 With Statement

with statements sesuai dengan konvensi pengkodean yang dibangun dalam Python yang disebut protokol manajemen konteks. with statement memungkinkan Anda untuk mengakses dan menggunakan objek yang memiliki fungsi _enter_ dan _exit_ tanpa perlu secara eksplisit menginisialisasi dan menghentikan objek tersebut [13].

III. Pembahasan

3.1 Analisis Program

```python
import pandas as pd
from functools import reduce

```

- Instal dan gunakan perpustakaan pandas untuk memanipulasi data dengan Python.
- Memuat fungsi pengurangan dari pustaka functools, yang digunakan untuk menerapkan fungsi pada data yang dapat diubah dan mengumpulkan hasilnya.

```python
# Baca file Excel
data = pd.read_csv('/content/911_Calls_for_Service_(Last_30_Days).csv')

# Filter baris dengan data yang hilang di kolom Zip atau Neighborhood
filtered_data = data.dropna(subset=['zip_code', 'neighborhood'])

# Konversi ke dalam bentuk daftar dictionary
filtered_dict = filtered_data.to_dict(orient='records')
```

- Baca file CSV bernama "911_Calls_for_Service_(Last_30_Days).csv" (seharusnya terletak di folder "/content") ke dalam data bernama Pandas DataFrame. Informasi dalam bingkai data ini dapat mencakup panggilan darurat (respon, waktu pengiriman, dan kemungkinan kondisi lingkungan).
- Buat DataFrame baru bernama filter_data dengan menghapus baris data yang nilainya hilang (diwakili oleh NaN) di kolom 'zip_code' atau 'neighborhood'. Hal ini memastikan analisis yang berpusat pada panggilan dengan data lokasi yang komprehensif.
- Konversikan Filter_ta DataFrame ke daftar kamus.Setiap kamus mewakili baris dalam DataFrame, dengan kunci yang sesuai dengan nama kolom dan nilai sebagai titik data yang sesuai.

```python
# Filter data untuk kepolisian Detroit
detroit_data = filter(lambda x: 'Detroit' in x['neighborhood'], filtered_dict)

# Fungsi untuk menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata
def calculate_averages(acc, curr):
    acc['total_response_time'] += curr['totalresponsetime']
    acc['total_dispatch_time'] += curr['dispatchtime']
    acc['total_time_on_scene'] += curr['time_on_scene']
    acc['count'] += 1
    return acc
```

- Buat iterator baru bernama detroit_data yang hanya berisi entri filter_dict dengan kunci "lingkungan" dalam kamus disetel ke "Detroit". Ini memisahkan panggilan khusus dari wilayah sekitar Detroit.

- Fungsi ini valid untuk dua argumen:
  acc: Kamus akumulator melacak total dan jumlah proses.
  curr: Kamus mewakili panggilan ke iterator detroit_data.n Tentukan jumlah
  total waktu respons untuk akumulator dan durasi panggilan serentaknya.
  Logika yang sama berlaku untuk total_dispatch_time dan
  total_time_on_scene.
  acc['count'] += 1 Meningkatkan jumlah panggilan yang ditangani.
  Kamus kumulatif yang diperbarui dikembalikan.

```
24 # Inisialisasi nilai awal
25 initial_values = {'total_response_time': 0, 'total_dispatch_time': 0, 'total_time_on_scene': 0, 'count': 0}
26
27 # Hitung total waktu respons, waktu pengiriman, dan total waktu rata-rata
28 totals = reduce(calculate_averages, detroit_data, initial_values)
29
30 # Hitung rata-rata
31 average_response_time = totals['total_response_time'] / totals['count']
32 average_dispatch_time = totals['total_dispatch_time'] / totals['count']
33 average_time_on_scene = totals['total_time_on_scene'] / totals['count']
```

- Buat kamus bernama init_values untuk menyimpan nilai awal akumulator. Total waktu diinisialisasi ke 0 dan hitungan diatur ke 0.
- Terapkan fungsi reduce. Ia mengulang entri detroit_data, meneruskan data dari akumulator dan panggilan saat ini ke fungsi Hitung_rata-rata. Fungsi ini memperbarui akumulator dengan rincian setiap panggilan. Nilai akumulasi terakhir disimpan dalam kamus penjumlahan.
- Hitung waktu respons rata-rata dengan membagi total waktu respons dengan jumlah panggilan (count).
- Perhitungan serupa dilakukan untuk Waktu_pengiriman Rata-rata dan Waktu_rata-rata di tempat kejadian untuk menemukan waktu pengiriman rata-rata dan waktu rata-rata di tempat kejadian untuk panggilan darurat di Detroit.

```
1 from functools import reduce
2
```

- Pustaka functools menyediakan fitur impor untuk fungsi reduce di baris ini. Data agregat akan dihitung berdasarkan neighborhood menggunakan fungsi ini nanti.

```python
# Fungsi untuk menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap neighborhood
def calculate_neighborhood_averages(acc, curr):
    neighborhood = curr[0]['neighborhood']
    total_response_time = sum(map(lambda x: x['totalresponsetime'], curr))
    total_dispatch_time = sum(map(lambda x: x['dispatchtime'], curr))
    total_time_on_scene = sum(map(lambda x: x['time_on_scene'], curr))
    count = len(curr)

    acc.append({
        'neighborhood': neighborhood,
        'total_response_time': total_response_time,
        'total_dispatch_time': total_dispatch_time,
        'total_time_on_scene': total_time_on_scene,
        'count': count,
        'average_response_time': total_response_time / count,
        'average_dispatch_time': total_dispatch_time / count,
        'average_time_on_scene': total_time_on_scene / count
    })
    return acc
```

- Fitur ini dapat menampung dua argumen
  acc: , yaitu daftar yang berisi hasil penghitungan setiap lingkungan.
  curr : data untuk lingkungan saat ini, sebagai daftar kamus.
- Maka fungsinya adalah:
  1. Dapatkan nama kabupaten dari data terkini terlebih dahulu.
  2. Hitung total waktu respons, waktu pengiriman, dan waktu di lokasi menggunakan fungsi lambda total dan anonim.
  3. Hitung jumlah data pada lingkungan ini menggunakan len.
  4. Membuat kamus baru yang berisi informasi tentang lingkungan, total durasi, jumlah data, dan durasi rata-rata untuk semua pengukuran tersebut.
  5. Tambahkan kamus baru ke akumulator (acc).
  6. Kirim ulang akumulator yang telah diperbarui.

```
# Bagi list dictionary menjadi list dictionary yang lebih kecil yang dipisahkan oleh neighborhood
neighborhood_data = map(lambda neighborhood: list(filter(lambda x: x['neighborhood'] == neighborhood, filtered_dict)), set(map(lambda x: x['neighborhood'], filtered_d

# Hitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap neighborhood
neighborhood_averages = reduce(calculate_neighborhood_averages, neighborhood_data, [])
```

- Dengan menggunakan fungsi map, baris ini dapat melakukan iterasi melalui neighborhood yang berbeda dengan kumpulan data yang berbeda. Selama iterasi, fungsi lambda anonim digunakan untuk memfilter data asli (filtered_dict) dan hanya memilih data yang lingkungannya cocok dengan lingkungan yang sedang diproses. Hasil pemfilteran kemudian dikelompokkan ke dalam daftar dan ditambahkan ke neighborhood_data.
- Reduce menggunakan fungsi hitung_neighborhood_averages dan melakukan iterasi melalui data lingkungan. Akumulator (acc) dimulai dari daftar kosong ([]). Fungsi hitung_neighborhood_averages dijalankan berulang kali untuk setiap lingkungan, dengan akumulator sebagai argumen pertama dan data lingkungan sebagai argumen kedua. Akumulator akan terus diperbarui dengan hasil perhitungan dari fungsi Hitung_Neighborhood_averages. Setelah iterasi selesai, neighborhood_averages akan berisi daftar kamus lengkap dengan durasi rata-rata untuk setiap lingkungan.

```
total_detroit_population = {
    'neighborhood': 'All Detroit',
    'total_response_time': totals['total_response_time'],
    'total_dispatch_time': totals['total_dispatch_time'],
    'total_time_on_scene': totals['total_time_on_scene'],
    'count': totals['count'],
    'average_response_time': average_response_time,
    'average_dispatch_time': average_dispatch_time,
    'average_time_on_scene': average_time_on_scene
}
neighborhood_averages.append(total_detroit_population)
```

- Baris ini menambahkan kamus total_detroit_population yang berisi semua data Detroit ke daftar Neighbor_averages.
- Saat dijalankan, neighboorhood_averages akan menjadi daftar kamus lengkap. Item pertama hingga terakhir 1 berisi data rata-rata berdasarkan lingkungan, sedangkan item terakhir berisi data total untuk seluruh Detroit.

```
1 import json
2
3 # Tulis data ke dalam file JSON
4 with open('output.json', 'w') as f:
5     json.dump(neighborhood_averages, f, indent=4)
6
```

- Kode ini akan menampilkan file JSON dengan data neighbour_averages sebagai properti file output.json.
- Dengan memanfaatkan file JSON ini, dimungkinkan untuk menyimpan hasil analisis data pada waktu respons, waktu pengiriman, dan waktu di lokasi untuk berbagai lingkungan di Detroit. Harap dicatat bahwa kode ini hanya melakukan penyimpanan data dan tidak melakukan analisis atau pemrosesan data lebih lanjut.

IV. Kesimpulan

Program ini menggunakan pandas untuk menganalisis data di Detroit, menggunakan file CSV bernama "911_Calls_for_Service_Last_30_Days.csv" di DataFrame. Data difilter menggunakan fungsi baru yang disebut filter_data, yang mengidentifikasi data yang tinggi dalam bidang tertentu. Data kemudian dikonversi menjadi file JSON menggunakan fungsi "reduce". Program ini menggunakan fungsi yang disebut "map" untuk memfilter data berdasarkan lingkungan, dan fungsi yang bernama "hitung_neighborhood_averages" untuk menghitung total populasi Detroit.

References

[1] . A. Kumar, "A Complete Guide to Pandas, from Installation to Advanced Data Analysis Techniques, 2nd Edition," in *Mastering Pandas*, Birmingham, Packt Publishing, 2019, pp. 127-130.

[2] . S. . K. Mukhiya and . U. Ahmed, "Perform EDA Techniques to Understand, Summarize, and Investigate Your Data," in *Hands-On Exploratory Data Analysis with Python*, Birmingham, Packt Publishing, 2020, p. 115.

[3] C. . Y. Cho, R. K. J. Tan, . J. A. A. Leong and A. S. Sidhu, in *Large Scale Data Analytics*, cham, Springer International Publishing, 2019, p. 44.

[4] J. Surya and Efitra , "DASAR-DASAR PEMROGRAMAN DENGAN PYTHON," Jambi, PT. Sonpedia Publishing Indonesia, 2023, p. 119.

[5] H. Balti and K. A. Weiss, "Job Ready Python," Hoboken, Wiley, 2021, pp. 49-52.

[6] S. F. Lott, "Functional Python Programming: Discover the power of functional programming,generator function. lazy evaluation,th ebuilt-in itertools library and monads," Birmingham Mumbai, Packet publishing Ltd., 2018, pp. 100-103.

[7] H. Wadi and ST, in *PEMROGRAMAN PYTHON : Untuk Pelajar & Mahasiswa*, Yogyakarta, TR Publisher, p. 67.

[8] M. R. faisal, D. Kartini, A. . R. Arrahimi and T. . H. Saragih, "Belajar Data Science: Text Mining Untuk Pemula I," BanjarBaru, Scripta Cendekia, 2023, p. 47.

[9] P. Miller and C. Bryce, "Leverage the Power of Python in Forensic Investigations, 2nd Edition," in *Learning Python for Forensics*, Birmingham, Packt Publishing, 2019, p. 133.

[10] A. Kadir, "Langkah Mudah Pemrograman OpenCV & Python," jakarta, Elex Media Komputindo, 2019, p. 56.

[11] H. Wadi and ST, in *PEMROGRAMAN PYTHON : Untuk Pelajar & Mahasiswa*, Yogyakarta, TR Publisher, p. 67.

[12] kahlil, M. . R. Munggaran, L. Kurnianggoro, . A. Mahendra, N. Zarima, F. Noviantika and A. Febriana, "Computer Vison Berbasis Deep Learning untuk Aplikasi Pertanian: Teori dan Praktik," Banda Aceh, Syiah Kuala University Press, 2023, p. 45.

[13] . P. Miller and C. Bryce, "Leverage the Power of Python in Forensic Investigations, 2nd Edition," in *Learning Python for Forensics*, Birmingham, Packt Publishing, 2019, p. 133.