Pendekatan Fungsional dalam Pengolahan Data 911 di Detroit

Oktavia Nurwinda Puspitasari[1], Mutiara Dian Pitaloka[2], Syadza Puspadari Azhar[3], Kayla Amanda Sukma[4], Muhammad Wafi Raihan[5]
Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera, Lampung Selatan, Indonesia
Email: oktavia.122450041@student.itera.ac.id mutiara.122450047@student.itera.ac.id syadza.122450072@student.itera.ac.id kayla.122450086@student.itera.ac.id muhammad.122450144@student.itera.ac.id

1. Pendahuluan

Salah satu layanan yang paling penting dikota mana pun adalah sistem panggilan darurat 911. Dimana sistem ini akan selalu dibutuhkan dan ditawarkan dalam bentuk bantuan selama keadaan darurat. Penanganan pada pengolahan data yang efektif dalam situasi ini penting untuk menjamin reaksi yang cepat dan efisien terhadap situasi darurat.

Kota di Detroit mengalami kelonjakan tingkat kejahatan, termasuk kekerasan, pencurian, mobil, dan perampokan yang menimbulkan kekhawatiran bagi masyarakat. Dalam upaya untuk meningkatkan pemrosesan data layanan darurat 911 agar lebih efisien menggunakan penerapan fungsi *map*, *filter,* dan *reduce* menjadi relevan. Fungsi-fungsi ini adalah bagian integral dari pemrograman fungsional yang memungkinkan manipulasi data dengan cara yang efisien dan efektif.

Penelitian ini berfokus pada penerapan *map*, *filter*, dan *reduce* dalam menangani data dari layanan panggilan darurat 911. Tujuan pada penelitian ini juga untuk menyederhanakan prosedur pemrosesan data, memfasilitasi analisis yang paling cepat dan secara signifikan meningkatkan respon sistem secara keseluruhan.

2. Metode

a. Pandas

Pandas merupakan *library open source* dalam bahasa pemrograman python. Pandas biasanya digunakan untuk melakukan analisis data serta memanipulasi data. *Library* pandas sangat berguna untuk melakukan *data wrangling* dan *data cleaning* sebelum data dilakukan analisis lebih lanjut. Hal tersebut supaya data menjadi lebih teratur dan bersih. Terdapat beberapa fitur utama yang tersedia pada *library* pandas yaitu *data frame*, series, manipulasi data, pembersihan data, integrasi dengan sumber data eksternal, dan analisis data. Pada kasus ini *library* pandas digunakan untuk membaca data dalam format CSV dan menyimpannya dalam bentuk *data frame* [1].

b. Reduce

Fungsi reduce diimport melalui modul functools. Fungsi reduce memiliki konsep yang mirip seperti fungsi map yaitu dapat dua argumen sekaligus yaitu fungsi dan iterable object. Dalam fungsi reduce, akan dihasilkan nilai kumulatif dari operasi fungsi masukan dibagi dengan nilai *iterable object* masukannya. Pada kasus ini fungsi reduce digunakan untuk menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata [2].

c. *Filter*

Dalam bahasa pemrograman python, filter digunakan untuk memilih elemen tertentu dalam suatu *iterable* berdasarkan kondisi tertentu. Filter dapat digunakan pada berbagai macam *iterable* dengan menggabungkannya dengan fungsi built-in maupun fungsi lambda. Secara keseluruhan, fungsi filter menggunakan fungsi sebagai argumen untuk melakukan filter pada elemen-elemen dalam suatu iterable object. Fungsi ini akan mengembalikan objek filter yang berisi elemen yang memenuhi kondisi yang telah ditentukan [3].

d. *Map*

Fungsi map adalah bagian dari fungsi *built-in function* dalam konsep *high order function* yang dapat menerapkan suatu fungsi pada banyak nilai secara efisien. Fungsi ini digunakan untuk membuat array baru dengan melakukan operasi pada setiap elemen dari array itu berasal. Dengan menggunakan map, dapat dihasilkan suatu koleksi baru dengan pengaplikasian elemen pada setiap objek yang dapat dilakukan iterasi seperti *range, list, tuple, dictionary,* dan lainnya. Maka dapat dikatakan bahwa fungsi map melakukan pemetaan data sehingga jumlah dan panjang data dari hasil map sama dengan *iterable* yang digunakan [4].

e. *Lambda*

Fungsi lambda disebut juga sebagai fungsi anonim adalah salah satu cara untuk membuat fungsi tanpa harus memberikan nama pada fungsi tersebut. Fungsi lambda dapat terdiri dari beberapa argumen dengan hanya satu ekspresi. Hal ini bertujuan supaya fungsi dapat disusun secara sederhana tanpa harus mendefinisikan secara terpisah. Selain itu juga, kata kunci "def" tidak perlu digunakan untuk mendefinisikan fungsi lambda. Dengan fungsi lambda, kompleksitas dalam penulisan kode dapat dikurangi dan memberikan fleksibilitas pada penulisan kode dalam bahasa pemrograman python [5].

f. *JSON*

JSON adalah singkatan dari *JavaScript Object Notation*. JSON merupakan format *file* yang berbasis teks yang digunakan pada proses pertukaran data antara web server dan penggunanya. JSON memiliki format penyimpanan informasi yang lebih terstruktur yang dipakai sebagai pengganti yang lebih ringkas dan ringan dari pada *Extensive Markup Language*. JSON akan mengubah bahasa pemrograman yang kompleks menjadi lebih ringkas supaya mudah untuk dibaca dan dipahami oleh penggunanya. Terdapat jenis-jenis value dalam JSON yaitu *object, array, string, boolean, null, number*. Pada kasus ini JSON digunakan untuk menulis data dalam file ".json" [6].

3. Hasil dan Pembahasan

```
import pandas as pd
from functools import reduce

data = pd.read_csv('911_Calls_for_Service_(Last_30_Days).csv')
```

Library yang akan digunakan dalam praktikum kali ini ialah *library* pandas dan functools. *Library* pandas digunakan dalam mengimport data dan membaca data yang akan digunakan dalam kasus kali ini sebelum melakukan pemrosesan data. Data yang kami gunakan berupa *file* Laporan Polisi Detroit dalam bentuk csv. Pada *library* functools, kami menggunakan *reduce* yang akan digunakan pada langkah setelahnya ketika akan menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata. Selain itu, fungsi ini juga digunakan dalam menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap *neighborhood*.

```
# 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')

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

Setelah itu, kita akan melakukan filter baris pada bagian data yang hilang pada kolom Zip atau *Neighborhood*. Kode ‘`data.dropna`‘ digunakan untuk menghapus baris-baris yang memiliki nilai ‘NaN’ atau nilai yang hilang pada kolom ‘zip_code’ atau ‘neighborhood’. Hasil ini akan disimpan dalam variabel `filtered_data`. Setelah di filter, kita akan mengkonversi variabel ‘`filtered_data`‘ menjadi bentuk *dictionary*, dimana setiap baris pada data menjadi sebuah *dictionary* dan setiap kolom akan menjadi kunci dalam *dictionary* tersebut. Hasil tersebut kemudian akan disimpan dalam variabel ‘`filtered_dict`‘ dimana setiap elemennya merupakan sebuah *dictionary* yang merepresentasikan satu baris dalam data. Setelah itu, kita akan melakukan filter untuk menyaring elemen-elemen dalam ‘`filtered_dict`’ apabila string ‘Detroit’ ada dalam nilai dari kunci ‘neighborhood’ pada setiap *dictionary* di dalamnya. Objek-objek filter yang telah di filter tersebut akan disimpan dalam variabel ‘`detroit_data`’.

```
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
```

Selanjutnya adalah menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata. Fungsi ini akan didefinisikan sebagai '`calculate_averages`' dengan parameter 'acc' dan 'curr' dimana 'acc' merupakan *dictionary* yang menyimpan total akumulasi dari beberapa metrik dan jumlah elemen yang telah di proses sedangkan 'curr' merupakan *dictionary* yang mewakili elemen data yang sedang diproses. Fungsi ini dapat digunakan bersamaan dengan fungsi *reduce* untuk menggabungkan elemen-elemen dalam sebuah list menjadi satu.

```
# Inisialisasi nilai awal
initial_values = {'total_response_time': 0, 'total_dispatch_time': 0,
'total_time_on_scene': 0, 'count': 0}

# Menghitung total waktu respons, waktu pengiriman, dan total waktu rata-
rata
totals = reduce(calculate_averages, detroit_data, initial_values)

# Menghitung rata-rata
average_response_time =  totals['total_response_time'] /
                         totals['count']
average_dispatch_time =  totals['total_dispatch_time'] /
                         totals['count']
average_time_on_scene =  totals['total_time_on_scene'] /
                         totals['count']
```

Pada kode di atas, kita akan menginisiasi nilai awal pada *dictionary*. Nilai dari masing-masing elemen dalam *dictionary* semuanya akan diinisiasikan ke 0. Setelah itu, kita akan menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata dengan menggunakan fungsi *reduce* menggunakan fungsi '`calculate_averages`' yang telah didefinisikan sebelumnya. Fungsi '`calculate_averages`' akan dijalankan secara iteratif pada setiap elemen dalam '`detroit_data`' dengan '`initial_values`' sebagai akumulator awal. Hasil *reduce* ini akan disimpan dalam variavel '`totals`' yang berisi *dictionary* dengan total jumlah dan jumlah elemen. Setelah itu, kita akan menghitung rata-rata waktu respons, waktu pengiriman, dan waktu di tempat kejadian.

```
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
```

Kode di atas mendefinisikan fungsi untuk menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap *neighborhood* dengan parameter 'acc' dan 'curr' dalam sekumpulan data. Fungsi ini akan mengakumulasi hasil per *neighborhood* dan menyimpan hasilnya dalam sebuah *dictionary*. Kode ini akan menghasilkan sebuah *dictionary* dimana setiap *dictionary* dalam daftar tersebut akan berisi informasi untuk satu *neighborhood*, termasuk total dan rata-rata dari tiga metrik waktu serta jumlah kejadian dalam *neighborhood* tersebut.

```
neighborhood_data = map(lambda neighborhood: list(filter(lambda x:
                        x['neighborhood'] == neighborhood,
                        filtered_dict)), set(map(lambda x:
                        x['neighborhood'], filtered_dict)))

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

Setelah itu, kita akan membuat list *dictionary* menjadi list *dictionary* yang lebih kecil yang dipisahkan oleh *neighborhood* dengan menggunakan fungsi map. Fungsi ini akan menghasilkan sebuah iterable berisi list-list *dictionary*, dimana setiap list *dictionary* hanya berisi data untuk satu *neighborhood* yang akan disimpan dalam variabel '`neighborhood_data`'. Setelah itu, kita akan menghitung total waktu respons, waktu pengiriman, dan total waktu rata-rata untuk setiap *neighborhood* dengan menggunakan fungsi *reduce*. Fungsi ini akan menghasilkan sebuah daftar *dictionary*, dimana setiap *dictionary* berisi informasi total dan rata-rata untuk satu *neighborhood*.

```
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)
```

Langkah terakhir ialah menambahkan item *dictionary* untuk menyertakan data populasi untuk semua Detroit. Setelah selesai menambahkan *dictionary*, kita akan menambahkan *dictionary* ke '`neighborhood_averages`'. Kode ini akan menambahkan ringkasan dari semua data kejadian di Detroit ke dalam daftar '`neighborhood_averages`' dengan membuat *dictionary* yang mewakili seluruh kota Detroit dan menggabungkannya dengan rata-rata total waktu dari semua kejadian di Detroit.

```
import json

with open('output.json', 'w') as f:
    json.dump(neighborhood_averages, f, indent=4)
```

Setelah selesai pada program, kita akan membuat daftar *dictionary* '`neighborhood_averages`' ke dalam sebuah file JSON bernama '`output.json`'. Hasil akhir dari kode ini ialah sebuah file yang berisi data untuk setiap *neighborhood* dan total untuk seluruh kota Detroit dalam format JSON.

4. Kesimpulan

Penelitian ini menerapkan teknik pemrograman fungsional menggunakan fungsi *map, filter,* dan *reduce* untuk meningkatkan efisiensi dan efektivitas dalam menangani panggilan darurat 911 di Kota Detroit. Dengan memanfaatkan *library* pandas dalam Python, pengolahan data panggilan darurat menjadi lebih terstruktur dan efisien.

Fungsi filter membantu memilih data berdasarkan kriteria tertentu. Fungsi map digunakan untuk mengoperasikan setiap elemen data. Sementara itu, fungsi reduce menggabungkan elemen-elemen data menjadi hasil akhir yang diinginkan. Pendekatan ini memungkinkan analisis data yang cepat dan akurat.

Hasil pengolahan data memberikan statistik penting seperti total waktu respons, rata-rata waktu respons, serta analisis berdasarkan wilayah di Detroit. Informasi ini membantu memahami pola kejadian darurat di kota secara lebih baik.

Akhirnya, penelitian menghasilkan file JSON yang berisi ringkasan data untuk setiap wilayah dan total keseluruhan Kota Detroit. Hal ini memberikan gambaran komprehensif tentang tingkat tanggap darurat di kota tersebut.

Secara keseluruhan, penerapan teknik pemrograman fungsional terbukti meningkatkan efisiensi operasional dan membantu dalam memahami serta mengelola situasi darurat dengan lebih baik di Kota Detroit.

5. Daftar Pustaka

[1] A. Ihwan, "Bahasa Python - Pandas Library," 12 July 2023. [Online]. Available: https://id.linkedin.com/pulse/bahasa-python-pandas-library-aris-ihwan. [Accessed 15 May 2024].

[2] A. "Menggunakan reduce (functools)," 22 June 2019. [Online]. Available: https://koding.alza.web.id/menggunakan-reduce-functools/. [Accessed 15 May 2024].

[3] A. Ihwan, "Bahasa Python - Filtering," 14 July 2023. [Online]. Available: https://id.linkedin.com/pulse/bahasa-python-filtering-aris-ihwan. [Accessed 15 May 2024].

[4] N. A. Prayogo, "Belajar Golang," 2009. [Online]. Available: https://dasarpemrogramangolang.novalagung.com/1-berkenalan-dengan-golang.html. [Accessed 15 May 2024].

[5] A. Muhardian, "Belajar Python: Membuat Fungsi dengan Lambda Expression," 18 December 2019. [Online]. Available: https://www.petanikode.com/python-lambda/. [Accessed 15 May 2024].

[6] F. A, "Apa Itu JSON? Penjelasan, Penggunaan, dan Contoh JSON," 7 December 2022. [Online]. Available: https://www.hostinger.co.id/tutorial/apa-itu-json. [Accessed 15 May 2024].

6. Lampiran

Link Colab = https://bit.ly/3wBCYJ5