Analisis Laporan Polisi Detroit Menggunakan Fungsi Map, Filter, dan Reduce

Randa Andriana Putra[1], Uliano Wilyam Purba[2], Tria Yunanni[3], Daris Samudra[4], Nasywa Nur Afifah[5]
122450083[1], 122450098[2], 122450062[3], 122450102[4], 122450125[5]
Program Studi Sains Data Institut Teknologi Sumatera
Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan, Lampung 35365
Email: randa.122450083@student.itera.ac.id[1], uliano.122450098@student.itera.ac.id[2], tria.122450062@student.itera.ac.id[3], daris.122450102@student.itera.ac.id[4], nasywa.122450125@student.itera.ac.id[5]

PENDAHULUAN

Analisis data operasional kepolisian merupakan elemen penting dalam upaya peningkatan kinerja dan efisiensi layanan kepolisian. Laporan ini memfokuskan pada pengolahan dan analisis data Laporan Polisi Detroit untuk menilai kinerja polisi, khususnya dalam hal waktu respons dan waktu pengiriman di berbagai neighborhood di kota tersebut. Melalui serangkaian langkah sistematis yang melibatkan teknik pemrograman berbasis fungsi tepatnya fungsi map, filter dan reduce laporan ini bertujuan memberikan gambaran yang komprehensif mengenai performa kepolisian Detroit.

Laporan ini memiliki relevansi tinggi dalam konteks peningkatan efisiensi dan efektivitas layanan kepolisian. Dengan menganalisis data waktu respons dan waktu pengiriman, laporan ini membantu mengidentifikasi kelemahan dalam kinerja kepolisian serta memberikan rekomendasi berbasis data untuk perbaikan layanan. Selain itu, metode analisis yang digunakan dalam penelitian ini, yang melibatkan teknik pemrograman modern seperti Map, Filter, dan Reduce, menunjukkan bagaimana teknologi dapat dioptimalkan untuk mengolah data besar dan kompleks. Pendekatan ini tidak hanya relevan bagi kepolisian Detroit, tetapi juga dapat diterapkan oleh departemen kepolisian lainnya yang ingin melakukan analisis serupa.

Dengan demikian, penelitian ini diharapkan dapat menjadi model bagi studi-studi berikutnya dalam bidang analisis data kepolisian, memberikan wawasan yang lebih baik tentang kinerja operasional, dan mendukung upaya peningkatan kualitas layanan kepolisian di berbagai kota.

METODE

A. Library CSV

Library csv dalam Python adalah alat yang ampuh untuk membaca, menulis, dan memanipulasi data dalam format csv. Format csv adalah format file teks biasa yang sering digunakan untuk menyimpan data tabular, seperti spreadsheet atau database.

Fitur utama library csv:

Membaca file csv, dapat digunakan untuk membaca file csv dan memuat datanya ke dalam struktur data python seperti daftar atau kamus. Menulis file csv, dapat digunakan untuk membuat file csv baru dan menulis data dari struktur data python ke dalamnya. Memanipulasi

data csv, dapat digunakan untuk memanipulasi data dalam file csv, seperti memfilter baris, menambahkan kolom, atau mengubah nilai. Mendukung berbagai dialek csv, memungkinkan untuk bekerja dengan berbagai format file csv. (Lutz, M. 2016)

B. Fungsi Map

Fungsi map() dalam Python adalah fungsi bawaan yang digunakan untuk menerapkan fungsi ke setiap elemen dalam suatu iterable (seperti daftar, tuple, string, dll.) dan menghasilkan iterable baru yang berisi hasil penerapan fungsi tersebut. Fungsi map() mengembalikan objek map yang merupakan iterator. Untuk mendapatkan hasil dari map(), perlu mengonversikannya menjadi daftar, tuple, atau iterable lainnya, misal dengan bantuan fungsi list() untuk mengkonversinya menjadi daftar atau list.

Kelebihan fungsi map:

Kode lebih singkat, membantu menulis kode yang lebih ringkas dan mudah dibaca, terutama untuk iterable besar. Efisiensi, bisa lebih efisien daripada loop for tradisional untuk iterable besar. Fleksibel, dapat bekerja dengan berbagai iterable dan fungsi. (Matthes, E. 2023)

C. Fungsi Filter

Fungsi filter() digunakan untuk menyaring (memfilter) elemen dalam suatu iterable berdasarkan kondisi tertentu dan menghasilkan iterable baru yang hanya berisi elemen yang memenuhi kondisi tersebut.

Cara Kerja:

1. Mengambil elemen dari iterable secara berurutan.
2. Mengevaluasi elemen dengan fungsi yang diberikan.
3. Jika fungsi mengembalikan True, simpan elemen dalam iterable baru.
4. Jika fungsi mengembalikan False, buang elemen.
5. Mengulangi langkah 1-4 hingga semua elemen dalam iterable telah diproses.

Fungsi filter() menawarkan beberapa keuntungan dalam memproses dan memanipulasi data, yaitu: memfokuskan pada elemen relevan, filter() membantu fokus pada elemen yang memenuhi kriteria tertentu, mengabaikan elemen yang tidak relevan. Hal ini memungkinkan untuk menganalisis dan memproses data yang lebih terarah dan efisien. Kode lebih singkat, filter() memungkinkan penulisan kode yang lebih ringkas dan mudah dibaca, terutama untuk menyaring iterable besar. Dapat menghindari loop for yang berulang dan rumit, menghasilkan kode yang lebih rapi dan mudah dipahami. Efisiensi, filter() dapat lebih efisien daripada loop for tradisional untuk menyaring iterable besar, terutama saat hanya perlu memproses sebagian kecil elemen.

Namun, penting untuk memahami beberapa kekurangan filter(), yaitu tidak mengubah iterable Asli, filter() tidak mengubah iterable asli. Iterable asli tetap sama setelah difilter. Jika ingin mengubah iterable asli, perlu menggunakan metode lain seperti list.sort() atau list.remove(). Hanya memfilter elemen, filter() hanya menyaring elemen. Jika perlu melakukan tindakan lain pada elemen yang difilter, seperti mengubah nilainya atau menambahkannya ke struktur data lain, perlu menggunakan loop for atau metode lain.

D. Fungsi Reduce

Fungsi reduce() dalam Python, yang juga dikenal sebagai functools.reduce(), digunakan untuk mengurangi elemen dalam suatu iterable menjadi satu nilai tunggal. Sintaks dasar reduce() adalah: reduce(function, iterable, initializer=None), function: Fungsi yang akan diterapkan pada elemen-elemen dalam iterable. Fungsi ini harus menerima dua argumen: nilai akumulasi saat ini dan elemen berikutnya dari iterable. iterable: Iterable yang berisi elemen yang akan direduksi. initializer (opsional): Nilai awal untuk akumulasi. Jika tidak disediakan, elemen pertama dari iterable akan digunakan sebagai nilai awal.

reduce() bekerja secara iteratif, menerapkan fungsi yang diberikan ke setiap pasangan elemen berturut-turut dalam iterable. Hasil dari setiap penerapan disimpan dalam nilai akumulasi. Pada akhir iterasi, nilai akumulasi akhir dikembalikan.

E. JSON

JSON (JavaScript Object Notation) adalah format data ringan dan mudah dibaca yang digunakan untuk pertukaran data. Format ini sering digunakan untuk komunikasi antara aplikasi web dan server, serta untuk menyimpan data dalam file. JSON mirip dengan format JavaScript, tetapi dapat digunakan dalam berbagai bahasa pemrograman, termasuk Python.

Data JSON tersusun atas dua struktur dasar:

Objek, kumpulan pasangan kunci-nilai, di mana kuncinya adalah string dan nilainya bisa berupa objek lain, array, string, angka, atau nilai boolean (benar atau salah). Array, urutan nilai yang dipisahkan koma, di mana setiap nilai bisa berupa objek, array, string, angka, atau nilai boolean. (Yana, H. 2014)

PEMBAHASAN

Sebelum melakukan analisis laporan polisi detroit, diperlukan beberapa library yaitu csv, reduce dari functools dan json. Library csv digunakan untuk membaca file dataset laporan polisi detroit yang bertipe file .csv. Library reduce digunakan untuk menerapkan fungsi pada seluruh elemen iterable, yang dapat berguna dalam beberapa tahap analisis, seperti penggabungan data atau perhitungan agregat. Sedangkan library json digunakan untuk membaca dan menulis data dalam format JSON, yang mungkin diperlukan jika data disimpan dalam format tersebut atau jika perlu berinteraksi dengan sistem lain yang menggunakan JSON sebagai format pertukaran data. Dengan memanfaatkan ketiga library ini, analisis laporan polisi Detroit dapat dilakukan dengan lebih efisien dan efektif, mulai dari membaca dan mempersiapkan data hingga menerapkan transformasi dan analisis yang diperlukan, serta menyimpan hasil analisis jika diperlukan.

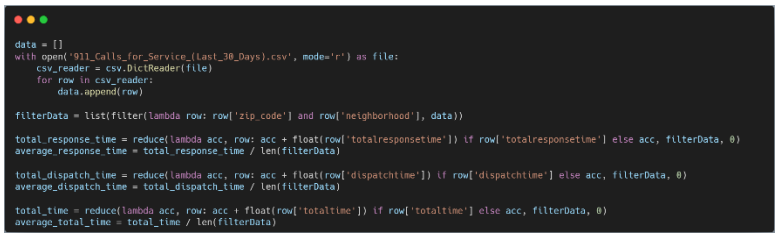
Pertama, data dari file CSV ('911_Calls_for_Service_(Last_30_Days).csv') dibaca menggunakan modul csv Python dengan mode pembacaan. Data tersebut kemudian dimasukkan ke dalam sebuah daftar untuk diproses lebih lanjut. Proses ini menggunakan

csv.DictReader untuk membaca setiap baris data sebagai kamus, di mana setiap kolom dalam file CSV akan menjadi kunci kamus.

Selanjutnya, data tersebut difilter untuk memastikan bahwa setiap entri memiliki nilai yang valid untuk kolom 'zip_code' dan 'neighborhood'. Hal ini dilakukan untuk memastikan bahwa hanya data yang lengkap dan relevan yang akan digunakan dalam analisis selanjutnya. Penggunaan fungsi filter() dengan ekspresi lambda yang sesuai memungkinkan penggunaan kriteria filter yang disesuaikan dengan kebutuhan analisis.

Setelah proses pemfilteran selesai, dilakukan perhitungan berbagai metrik yang relevan untuk mengevaluasi kinerja layanan 911 selama periode waktu tersebut. Pertama-tama, total waktu respon, total waktu pengiriman, dan total waktu panggilan untuk semua entri yang lolos filter tersebut dihitung menggunakan fungsi reduce() dari modul functools. Fungsi ini menerapkan operasi penjumlahan pada setiap elemen data untuk mengumpulkan total waktu dari masing-masing kategori.

Setelah total waktu dihitung, dilakukan perhitungan rata-rata waktu respon, rata-rata waktu pengiriman, dan rata-rata waktu panggilan dengan membagi total waktu masing-masing kategori dengan jumlah entri dalam data yang telah difilter. Ini memberikan gambaran yang lebih rinci tentang kinerja layanan 911 selama periode 30 hari terakhir di Detroit.



```
data = []
with open('911_Calls_for_Service_(Last_30_Days).csv', mode='r') as file:
    csv_reader = csv.DictReader(file)
    for row in csv_reader:
        data.append(row)

filterData = list(filter(lambda row: row['zip_code'] and row['neighborhood'], data))

total_response_time = reduce(lambda acc, row: acc + float(row['totalresponsetime']) if row['totalresponsetime'] else acc, filterData, 0)
average_response_time = total_response_time / len(filterData)

total_dispatch_time = reduce(lambda acc, row: acc + float(row['dispatchtime']) if row['dispatchtime'] else acc, filterData, 0)
average_dispatch_time = total_dispatch_time / len(filterData)

total_time = reduce(lambda acc, row: acc + float(row['totaltime']) if row['totaltime'] else acc, filterData, 0)
average_total_time = total_time / len(filterData)
```

Setelah memfilter data berdasarkan nama-nama lingkungan (neighborhood), langkah berikutnya adalah menghitung statistik rata-rata waktu respons, waktu pengiriman, dan waktu total panggilan untuk setiap lingkungan. Proses ini dilakukan dengan iterasi melalui setiap nama lingkungan unik yang ada dalam data yang telah difilter sebelumnya. Setiap lingkungan kemudian diambil sebagai basis untuk memfilter data, sehingga hanya entri yang terkait dengan lingkungan tersebut yang akan diproses dalam perhitungan statistik.

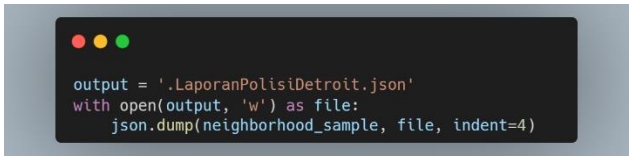
Dalam iterasi tersebut, total waktu respons, total waktu pengiriman, dan total waktu panggilan untuk setiap lingkungan dihitung menggunakan fungsi reduce() dari modul functools, seperti yang dilakukan sebelumnya. Setelah itu, rata-rata waktu respons, waktu pengiriman, dan waktu total panggilan untuk setiap lingkungan dihitung dengan membagi total waktu masing-masing kategori dengan jumlah entri yang terkait dengan lingkungan tersebut.

Hasil dari perhitungan statistik untuk setiap lingkungan disimpan dalam sebuah daftar neighborhood_sample sebagai kamus, di mana setiap kamus berisi informasi tentang nama lingkungan, rata-rata waktu respons, rata-rata waktu pengiriman, dan rata-rata waktu total panggilan untuk lingkungan tersebut.

Selain itu, statistik rata-rata waktu respons, waktu pengiriman, dan waktu total panggilan untuk seluruh Detroit juga dihitung dan disimpan dalam sebuah kamus detroit_data. Informasi ini juga dimasukkan ke dalam daftar neighborhood_sample, sehingga informasi tentang seluruh Detroit dan setiap lingkungan tersedia dalam struktur data yang sama.

```
neighborhoods = list(set(map(lambda row: row['neighborhood'], filterData)))

neighborhood_sample = []

for neighborhood in neighborhoods:
    neighborhood_data = list(filter(lambda row: row['neighborhood'] == neighborhood, filterData))

    total_response_time_neigh = reduce(lambda acc, row: acc + float(row['totalresponsetime']) if row['totalresponsetime'] else acc,
neighborhood_data, 0)
    average_response_time_neigh = total_response_time_neigh / len(neighborhood_data)

    total_dispatch_time_neigh = reduce(lambda acc, row: acc + float(row['dispatchtime']) if row['dispatchtime'] else acc, neighborhood_data, 0)
    average_dispatch_time_neigh = total_dispatch_time_neigh / len(neighborhood_data)

    total_time_neigh = reduce(lambda acc, row: acc + float(row['totaltime']) if row['totaltime'] else acc, neighborhood_data, 0)
    average_total_time_neigh = total_time_neigh / len(neighborhood_data)

    neighborhood_sample.append({
        'neighborhood': neighborhood,
        'average_response_time': average_response_time_neigh,
        'average_dispatch_time': average_dispatch_time_neigh,
        'average_total_time': average_total_time_neigh
    })

detroit_data = {
    'neighborhood': 'All Detroit',
    'average_response_time': average_response_time,
    'average_dispatch_time': average_dispatch_time,
    'average_total_time': average_total_time
}
neighborhood_sample.append(detroit_data)
```

Terakhir, simpan hasil analisis dalam format json dengan menggunakan fungsi json.dump() untuk menuliskan data dari variabel neighborhood_sample ke dalam file dengan nama yang telah ditentukan (LaporanPolisiDetroit.json). Selain itu, parameter indent=4 digunakan untuk memformat output JSON dengan indentasi agar lebih mudah dibaca.



KESIMPULAN

Analisis data operasional kepolisian terhadap Laporan Polisi Detroit menggunakan teknik pemrograman berbasis fungsi, seperti map, filter, dan reduce, memberikan gambaran komprehensif tentang kinerja kepolisian kota tersebut. Dengan fokus pada waktu respons dan pengiriman di berbagai lingkungan, analisis ini tidak hanya membantu mengidentifikasi kelemahan dalam kinerja layanan kepolisian, tetapi juga memberikan rekomendasi berbasis data untuk perbaikan layanan. Melalui penyimpanan hasil analisis dalam format JSON, informasi yang relevan dapat mudah diakses dan digunakan untuk membuat keputusan yang tepat dalam upaya meningkatkan responsibilitas layanan darurat. Sebagai hasilnya, penelitian ini diharapkan dapat menjadi model bagi studi-studi berikutnya dalam bidang analisis data kepolisian, memberikan wawasan yang lebih baik tentang kinerja operasional, dan mendukung upaya peningkatan kualitas layanan kepolisian di berbagai kota.

REFERENSI

Lutz, M. (2016). Dive Into Python 3. O'Reilly Media.

Matthes, E. (2023). Python 3 for Everybody (2nd ed.). No Starch Press.

Yana, H. (2014). APLIKASI RUMUS MATEMATIKA SMA BERBASIS MOBILE. Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)