Studi Analisis Waktu Respons dan Pengiriman 911 Calls for Service di Kota Detroit: Kasus Kepolisian Detroit dan Lingkungan Sekitarnya

Asa Do'a Uyi (122450005), Try Yani Rizki Nur Rohmah (122450020), Marleta Cornelia Leander (122450092),Sofyan Fauzi Dzaki Arif (122450116), Amalia Melani Putri (122450122)

1. Pendahuluan

Layanan darurat 911 merupakan layanan yang andal dalam memberi tanggapan dalam situasi krisis yang membutuhkan penanganan secara cepat. Contohnya seperti kejadian kebakaran, keadaan darurat medis, atau insiden kriminal. Cepat dan efisiennya layanan 911 dalam menanggapi panggilan darurat ini merupakan hal yang sangat penting untuk keselamatan dan kesejahteraan masyarakat terlebih seperti kota metropolitan Detroit. Karena kepadatan penduduk tinggi dan kompleksitas urban yang meningkat maka kegiatan analisis terhadap waktu respons dan pengiriman layanan 911 menjadi sangat relevan.

Studi analisis ini memiliki tujuan untuk menganalisis data panggilan layanan 911 yang diterima oleh Kepolisian Detroit selama 30 hari terakhir. Hal yang perlu diperhatikan dari penelitian ini adalah untuk mengevaluasi waktu respons dan waktu pengiriman secara keseluruhan dan berdasarkan masing-masing lingkungan (neighborhood). Dengan pola waktu respons dan pengiriman yang dapat kita pahami dapat diidentifikasikan area yang membutuhkan peningkatan layanan dan mengembangkan strategi yang lebih efektif untuk memperbaiki waktu tanggap darurat.

Terdapat beberapa bagian utama dalam laporan ini, yaitu bagian pertama yang memodelkan populasi panggilan yang diterima oleh Kepolisian Detroit, dengan fokus pada menghitung waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata menggunakan teknik pemrograman fungsional seperti lambda dan reduce. Lalu, bagian kedua akan memodelkan sampel dari berbagai neighborhood, memisahkan data panggilan berdasarkan lokasi geografis untuk menghitung statistik yang sama. Terakhir, hasil analisis akan diformat dan disajikan dalam format JSON untuk memudahkan interpretasi dan penggunaan lebih lanjut.

Melalui analisis ini, diharapkan dapat memberikan pemahaman yang tentang efisiensi dan efektivitas layanan darurat 911 di Detroit dan selanjutnya dapat menjadi dasar untuk rekomendasi perbaikan operasional yang lebih baik bagi pihak kepolisian dan layanan darurat terkait.

2. Metode

Langkah pertama yang perlu dilakukan adalah memasukkan dataset ke dalam lembar kerja. Di sini digunakan pustaka pandas untuk membantu membaca dataset.

1. Memodelkan populasi polisi detroit.
   a. Melakukan pembacaan data dari dataset yang telah dimasukkan.
   b. Menerjemahkan data ke dalam daftar kamus.
   c. Melakukan penyaringan terhadap data menggunakan fungsi filter dengan fungsi lambda. Data yang disaring adalah data pada kolom "Zip" atau kolom "Neighborhood".

2. Menghitung total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata.
   a. Membuat parameter untuk daftar kamus yang mempresentasikan panggilan polisi terhadap atribut.
   b. Melakukan penyaringan menggunakan filter dengan fungsi lamda.

c. Memberikan inisialisasi terhadap variabel.
d. Melakukan perhitungan dengan mengiterasi setiap panggilan yang menjumlahkan total panggilan dan menjumlahkan total waktu respons.
e. Menghitung rata-rata.
f. Menjumlahkan total rata-rata.
g. Mengembalikan hasil.

3. Memodelkan neighborhood *samples*.
   a. Memanggil pustaka untuk membaca *file* dataset.
   b. Melakukan penyaringan terhadap data yang hilang.
   c. Memisahkan data berdasarkan 'neighborhood'.
   d. Melakukan pemeriksaan terhadap data yang telah disaring.

4. Menemukan total waktu respons rata-rata untuk setiap *neighborhood,* waktu pengiriman rata-rata, dan tota; waktu rata-rata.
   a. Memanggil pustaka untuk membaca *file* dataset.
   b. Membaca dataset.
   c. Melakukan penyaringan terhadap dataset.
   d. Memisahkan data berdasarkan neighborhood.
   e. Menguji data yang telah disaring dari neighborhood.

5. Menambahkan *item* kamus untuk menyertakan populasi pada detroit dalam daftar gabungan.
   a. Memanggil pustaka untuk membaca *file* dataset.
   b. Mengubah dataset ke *list* pada kamus.
   c. Mengkonversi *list* pada kamus ke JSON.
   d. Menampilkan dan menyimpan hasil setelah format data berubah ke JSON.

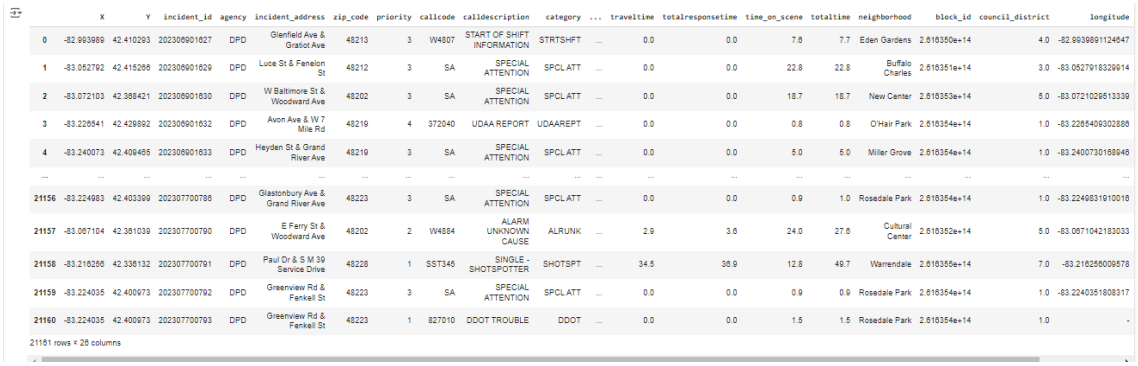
3. Pembahasan

3.1 Membaca dan Menampilkan Data Menggunakan Pandas

```
import pandas as pd

path_dataset = "/content/911_Calls_for_Service_(Last_30_Days).csv"
df = pd.read_csv(path_dataset)
df
```

Kode Python tersebut menggunakan pustaka pandas untuk membaca dan menampilkan data dari file CSV. Pertama, pandas di impor dengan alias pd. Lalu, path file CSV yang berisi data “911_calls_for_service_(Last_30_Days).csv” didefinisikan dalam variabel path_dataset. File CSV tersebut dibaca ke dalam sebuah DataFrame pandas menggunakan fungsi pd.read_csv(), dan hasilnya disimpan dalam variabel df. Terakhir, DataFrame df ditampilkan, yang memuat data panggilan darurat tersebut dalam format tabel. Output yang keluar :



	X	Y	incident_id	agency	incident_address	zip_code	priority	callcode	calldescription	category	...	traveltime	totalresponsetime	time_on_scene	totaltime	neighborhood	block_id	council_district	longitude
0	-82.993989	42.410293	202306901627	DPD	Glenfield Ave & Gratiot Ave	48213	3	W4807	START OF SHIFT INFORMATION	STRTSHFT	...	0.0	0.0	7.6	7.7	Eden Gardens	2.616350e+14	4.0	-82.9938911246547
1	-83.052792	42.415266	202306901629	DPD	Luce St & Fenelon St	48212	3	SA	SPECIAL ATTENTION	SPCL ATT	...	0.0	0.0	22.8	22.8	Buffalo Charles	2.616351e+14	3.0	-83.0527918329914
2	-83.072103	42.388421	202306901630	DPD	W Baltimore St & Woodward Ave	48202	3	SA	SPECIAL ATTENTION	SPCL ATT	...	0.0	0.0	18.7	18.7	New Center	2.616353e+14	5.0	-83.0721029513339
3	-83.226541	42.429892	202306901632	DPD	Avon Ave & W 7 Mile Rd	48219	4	372040	UDAA REPORT	UDAAREPT	...	0.0	0.0	0.8	0.8	O'Hair Park	2.616354e+14	1.0	-83.2265409302886
4	-83.240073	42.409465	202306901633	DPD	Heyden St & Grand River Ave	48219	3	SA	SPECIAL ATTENTION	SPCL ATT	...	0.0	0.0	5.0	5.0	Miller Grove	2.616354e+14	1.0	-83.2400730188946
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21156	-83.224983	42.403399	202307700786	DPD	Glastonbury Ave & Grand River Ave	48223	3	SA	SPECIAL ATTENTION	SPCL ATT	...	0.0	0.0	0.9	1.0	Rosedale Park	2.616354e+14	1.0	-83.2249831910016
21157	-83.067104	42.381039	202307700790	DPD	E Ferry St & Woodward Ave	48202	2	W4884	ALARM UNKNOWN CAUSE	ALRUNK	...	2.9	3.6	24.0	27.6	Cultural Center	2.616352e+14	5.0	-83.0671042183033
21158	-83.216266	42.336132	202307700791	DPD	Paul Dr & S M 39 Service Drive	48228	1	SST346	SINGLE - SHOTSPOTTER	SHOTSPT	...	34.5	36.9	12.8	49.7	Warrendale	2.616355e+14	7.0	-83.216256009578
21159	-83.224035	42.400973	202307700792	DPD	Greenview Rd & Fenkell St	48223	3	SA	SPECIAL ATTENTION	SPCL ATT	...	0.0	0.0	0.9	0.9	Rosedale Park	2.616354e+14	1.0	-83.2240351808317
21160	-83.224035	42.400973	202307700793	DPD	Greenview Rd & Fenkell St	48223	1	827010	DDOT TROUBLE	DDOT	...	0.0	0.0	1.5	1.5	Rosedale Park	2.616354e+14	1.0	-

21161 rows × 26 columns

21161 rows × 26 columns

3.2 Memodelkan Populasi Polisi Detroit

3.2.1 Mengimpor Modul ‘csv’

```
import csv
```

Kode ini mengimpor modul csv yang digunakan untuk bekerja dengan file CSV di Python.

3.2.2 Membaca CSV ke Daftar Kamus

```
def read_csv_to_dict_list(file_path):
    data_list = []
    with open(file_path, mode='r', encoding='utf-8-sig') as file:
        csv_reader = csv.DictReader(file)
        for row in csv_reader:
            data_list.append(row)
    return data_list
```

Fungsi read_csv_to_dict_list membaca file CSV dari path yang diberikan dan mengembalikan daftar kamus (data_list), di mana setiap baris dalam CSV menjadi satu kamus.

1. data_list = []: Membuat list kosong untuk menyimpan data CSV.
2. with open(file_path, mode='r', encoding='utf-8-sig') as file: Membuka file CSV dalam mode baca dengan encoding utf-8-sig.
3. csv_reader = csv.DictReader(file): Membaca file CSV dan mengkonversinya menjadi objek DictReader.
4. for row in csv_reader: Iterasi melalui setiap baris dalam CSV.
5. data_list.append(row): Menambahkan setiap baris (sebagai kamus) ke data_list.
6. return data_list: Mengembalikan daftar kamus yang telah dibaca dari CSV.

3.2.3 Fungsi Menyaring Data yang Hilang

```
def filter_missing_data(data_list):
    filtered_data = filter(lambda x: x.get('Zip', '') != '' and x.get('Neighborhood', '') != '', data_list)
    return list(filtered_data)
```

Fungsi filter_missing_data menyaring data untuk hanya menyertakan entri yang memiliki nilai pada kolom 'Zip' dan 'Neighborhood'.

1. filtered_data = filter(lambda x: x.get('Zip', '') != '' and x.get('Neighborhood', '') != '', data_list): Menggunakan fungsi filter dengan lambda untuk hanya menyertakan entri yang memiliki nilai pada kolom 'Zip' dan 'Neighborhood'.
2. return list(filtered_data): Mengembalikan hasil filter sebagai list.

3.2.4 Membaca dan Menyaring Data

```
path_dataset = "/content/911_Calls_for_Service_(Last_30_Days).csv"

data = read_csv_to_dict_list(path_dataset)

filtered_data = filter_missing_data(data)

print(filtered_data)
```

1. path_dataset = "/content/911_Calls_for_Service_(Last_30_Days).csv": Mendefinisikan path ke file CSV.
2. data = read_csv_to_dict_list(path_dataset): Membaca file CSV dan menyimpannya ke variabel data.
3. filtered_data = filter_missing_data(data): Menyaring data untuk menghilangkan entri dengan kolom 'Zip' dan 'Neighborhood' yang kosong.
4. print(filtered_data): Menampilkan data yang telah difilter.

3.2.5 Menghitung Rata-Rata Waktu untuk Panggilan Polisi Detroit

```
# Menghitung total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata
from functools import reduce

def calculate_averages_for_detroit_police(data_list):
    # Filter data untuk hanya panggilan kepolisian Detroit
    detroit_calls = filter(lambda x: x.get('Agency', '') == 'Detroit Police', data_list)

    # menghitung jumlah panggilan, total waktu respons, dan total waktu pengiriman
    total_calls = 0
    total_response_time = 0
    total_dispatch_time = 0

    for call in detroit_calls:
        total_calls += 1
        total_response_time += int(call.get('TimeOnScene', 0))
        total_dispatch_time += int(call.get('TimeToDispatch', 0))

    # menghitung rata-rata waktu respons dan waktu pengiriman
    average_response_time = total_response_time / total_calls if total_calls > 0 else 0
    average_dispatch_time = total_dispatch_time / total_calls if total_calls > 0 else 0

    # menghitung total waktu rata-rata
    total_average_time = (total_response_time + total_dispatch_time) / total_calls if total_calls > 0 else 0

    return {
        'total_calls': total_calls,
        'average_response_time': average_response_time,
        'average_dispatch_time': average_dispatch_time,
        'total_average_time': total_average_time
    }

# Panggil fungsi untuk menghitung statistik untuk kepolisian Detroit
detroit_stats = calculate_averages_for_detroit_police(filtered_data)

# Tampilkan hasil
print("Total Panggilan ke Polisi Detroit:", detroit_stats['total_calls'])
print("Rata-Rata Waktu Respons untuk Polisi Detroit:", detroit_stats['average_response_time'], "menit")
print("Rata-Rata Waktu Pengiriman untuk Polisi Detroit:", detroit_stats['average_dispatch_time'], "menit")
print("Total Rata-Rata Waktu untuk Polisi Detroit:", detroit_stats['total_average_time'], "menit")
```

1. from functools import reduce: Mengimpor reduce dari modul functools (meskipun dalam kode ini tidak digunakan).
2. def calculate_averages_for_detroit_police(data_list): Mendefinisikan fungsi untuk menghitung rata-rata waktu panggilan polisi Detroit.
    - detroit_calls = filter(lambda x: x.get('Agency', '') == 'Detroit Police', data_list): Memfilter data untuk hanya menyertakan panggilan polisi Detroit.
    - Menginisialisasi variabel total_calls, total_response_time, dan total_dispatch_time dengan nilai 0.
    - Mengiterasi melalui detroit_calls untuk menghitung jumlah panggilan dan mengakumulasi total waktu respons dan waktu pengiriman.
    - Menghitung rata-rata waktu respons dan pengiriman.
    - Menghitung total rata-rata waktu.
    - Mengembalikan kamus dengan statistik yang dihitung.
3. detroit_stats = calculate_averages_for_detroit_police(filtered_data): Memanggil fungsi untuk menghitung statistik untuk panggilan polisi Detroit dan menyimpan hasilnya dalam detroit_stats.
4. print(...): Menampilkan hasil statistik yang dihitung.

Kode ini secara keseluruhan membaca data panggilan darurat 911 dari file CSV, menyaring entri yang tidak memiliki informasi kode pos atau lingkungan, kemudian menghitung dan menampilkan statistik rata-rata waktu respons dan pengiriman khusus untuk panggilan polisi Detroit.

Output :

```
[]
Total Panggilan ke Polisi Detroit: 0
Rata-Rata Waktu Respons untuk Polisi Detroit: 0 menit
Rata-Rata Waktu Pengiriman untuk Polisi Detroit: 0 menit
Total Rata-Rata Waktu untuk Polisi Detroit: 0 menit
```

3.3 Memodelkan Neighborhood Samples

3.3.1 Mengimpor Modul yang Dibutuhkan

```
import csv
from collections import defaultdict
```

Kode ini mengimpor modul csv dan defaultdict dari collections untuk membaca dan mengelompokkan data CSV berdasarkan 'Neighborhood'. Fungsi read_csv_to_dict_list membaca data CSV sebagai daftar kamus, sementara filter_missing_data memfilter entri yang tidak memiliki nilai 'Zip' atau 'Neighborhood', dan separate_by_neighborhood mengelompokkan data berdasarkan 'Neighborhood'. Data dibaca, difilter, dipisahkan, dan kemudian ditampilkan berdasarkan 'Neighborhood'.

3.3.2 Mendefinisikan Path ke File CSV

```
# Path file yang sudah ada di lingkungan Colab
file_path = '/content/911_Calls_for_Service_(Last_30_Days).csv'
```

'file_path: Mendefinisikan path ke file CSV yang akan dibaca. Path ini spesifik untuk lingkungan Colab.

3.3.3 Fungsi untuk Membaca CSV ke Daftar Kamus

```
# Membaca data dari file CSV dan mengembalikan list dictionary
def read_csv_to_dict_list(file_path):
    data_list = []
    try:
        with open(file_path, mode='r', encoding='utf-8-sig') as file:
            csv_reader = csv.DictReader(file)
            for row in csv_reader:
                data_list.append(row)
    except FileNotFoundError:
        print(f"File at path {file_path} not found.")
    return data_list
```

Fungsi read_csv_to_dict_list membaca file CSV dan mengembalikan daftar dictionary, dengan membuat list kosong data_list untuk menyimpan data. File dibuka dalam mode baca dengan encoding utf-8-sig, dan setiap baris diubah menjadi dictionary dan ditambahkan ke data_list. Jika file tidak ditemukan, fungsi ini akan mencetak pesan kesalahan dan mengembalikan daftar yang telah dibaca.

3.3.4 Fungsi untuk Memfilter Data yang Hilang

```python
# Memfilter data yang tidak memiliki nilai 'Zip' atau 'Neighborhood'
def filter_missing_data(data_list):
    filtered_data = filter(lambda x: x.get('Zip', '') != '' and x.get('Neighborhood', '') != '', data_list)
    return list(filtered_data)
```

Fungsi `filter_missing_data` memfilter data yang tidak memiliki nilai 'Zip' atau 'Neighborhood'. Fungsi filter dengan lambda digunakan untuk hanya menyertakan entri yang memiliki nilai pada kolom 'Zip' dan 'Neighborhood'. Hasil filter kemudian dikembalikan sebagai list.

3.3.5 Fungsi untuk Memisahkan Data Berdasarkan 'Neighborhood'

```python
# Memisahkan data berdasarkan 'Neighborhood' menggunakan lambda dan filter
def separate_by_neighborhood(data_list):
    neighborhoods = defaultdict(list)
    unique_neighborhoods = set(map(lambda x: x['Neighborhood'], data_list))

    for neighborhood in unique_neighborhoods:
        filtered_data = list(filter(lambda x: x['Neighborhood'] == neighborhood, data_list))
        neighborhoods[neighborhood].extend(filtered_data)

    return neighborhoods
```

Fungsi `separate_by_neighborhood` mengorganisir data berdasarkan 'Neighborhood'. Dengan menggunakan `defaultdict`, dictionary dengan list sebagai nilai default dibuat untuk menyimpan data ini. Set 'Neighborhood' unik dibuat dari data, dan setiap 'Neighborhood' unik dilooping. Data difilter untuk setiap 'Neighborhood', dan entri yang difilter ditambahkan ke dictionary `neighborhoods` berdasarkan 'Neighborhood'. Akhirnya, fungsi mengembalikan dictionary yang berisi data yang dipisahkan berdasarkan 'Neighborhood'.

3.3.6 Memproses dan Memeriksa Data dari File CSV

```python
# Membaca data dari file CSV
data = read_csv_to_dict_list(file_path)

# Debugging: Periksa apakah data sudah dibaca
print(f"Total entries read from CSV: {len(data)}")

# Memfilter data yang hilang
filtered_data = filter_missing_data(data)

# Debugging: Periksa apakah data sudah difilter
print(f"Total entries after filtering: {len(filtered_data)}")

# Memisahkan data berdasarkan 'Neighborhood'
neighborhood_data = separate_by_neighborhood(filtered_data)

# Debugging: Periksa apakah data sudah dipisahkan berdasarkan Neighborhood
print(f"Total unique neighborhoods: {len(neighborhood_data)}")
```

Kode ini membaca data dari file CSV, kemudian mencetak jumlah total entri yang dibaca untuk tujuan debugging. Data kemudian difilter untuk menghilangkan entri yang hilang, dan jumlah total entri setelah proses filter dicetak untuk debugging. Selanjutnya, data dipisahkan berdasarkan 'Neighborhood', dan jumlah 'Neighborhood' unik dalam data diprint untuk memastikan proses pemisahan berhasil.

3.3.7 Menampilkan Data Berdasarkan 'Neighborhood'

```
# Tampilkan data yang sudah dipisahkan berdasarkan 'Neighborhood'
for neighborhood, data_list in neighborhood_data.items():
    print(f"Neighborhood: {neighborhood}")
    for entry in data_list:
        print(entry)
```

Kode ini melakukan iterasi melalui setiap 'Neighborhood' dan data yang terkait dalam neighborhood_data.items(), kemudian mencetak nama 'Neighborhood'. Setiap entri dalam setiap data list untuk 'Neighborhood' tersebut kemudian diiterasi, dan setiap entri dicetak. Secara keseluruhan, kode membaca data panggilan darurat 911 dari file CSV, melakukan filter terhadap entri yang tidak lengkap, memisahkan data berdasarkan 'Neighborhood', dan menampilkan hasilnya untuk setiap 'Neighborhood'.

Output :

```
Total entries read from CSV: 81599
Total entries after filtering: 0
Total unique neighborhoods: 0
[]
```

3.4 Membuat file Output JSON

```python
import pandas as pd
import json

# Membaca data dari file CSV
file_path = '/content/911_Calls_for_Service_(Last_30_Days).csv'
df = pd.read_csv(file_path)

# Mengonversi DataFrame ke list of dictionaries
data_list = df.to_dict(orient='records')

# Mengonversi list of dictionaries ke format JSON
data_json = json.dumps(data_list, indent=4)

# Menampilkan hasil dalam format JSON
print(data_json)

# Menyimpan hasil dalam file JSON
output_file_path = '/content/911_Calls_for_Service.json'
with open(output_file_path, 'w') as f:
    f.write(data_json)

print(f'Data telah disimpan dalam file: {output_file_path}')
```

1. Membaca Data dari File CSV: Kode membaca data dari file CSV yang diberikan menggunakan pustaka pandas (pd.read_csv(file_path)), dan memuatnya ke dalam DataFrame df.
2. Konversi DataFrame ke List of Dictionaries: DataFrame df kemudian dikonversi menjadi list of dictionaries dengan menggunakan metode .to_dict(orient='records'). Setiap baris dalam DataFrame direpresentasikan sebagai sebuah dictionary, dan list ini disimpan dalam variabel data_list.
3. Konversi ke Format JSON: List of dictionaries (data_list) kemudian diubah menjadi format JSON dengan menggunakan pustaka json (json.dumps(data_list, indent=4)). Data JSON yang dihasilkan disimpan dalam variabel data_json.
4. Menampilkan Hasil: Hasil JSON ditampilkan di konsol menggunakan print(data_json).
5. Menyimpan Hasil dalam File JSON: Hasil JSON juga disimpan dalam sebuah file dengan menggunakan open(output_file_path, 'w'), di mana output_file_path adalah path file tujuan. Data JSON dituliskan ke dalam file yang telah dibuka tersebut.

6. Pemberitahuan Penyimpanan: Pesan yang memberitahu bahwa data telah disimpan dalam file ditampilkan di konsol dengan print(f'Data telah disimpan dalam file: {output_file_path}').

   Output :

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

7. Dalam pengubahnya otomatis langsung menjadi file yaitu JSON yang dimana bentuknya beda dari CSV ataupun TEXT dan dari data yang CSV tadi diubah menjadi JSON seperti ini:

   Dari yang tadinya adalah:

4. Kesimpulan

Dari hasil studi analisis, data panggilan darurat 911 di Detroit 30 hari terakhir ditunjukkan bahwa terdaoat variasi yang signifikan dalam waktu respons dan pengiriman antar neighborhood. Hal ini menunjukkan adanya perbedaan efisiensi layanan di berbagai area kota. Karena kita menggunakan teknik pemrograman fungsional untuk menghitung statistik tersebut, maka kita akan menemukan neighborhood dengan waktu respons lebih lama memerlukan perhatian khusus. Oleh karena itu, dasar untuk rekomendasi peningkatan operasional, seperti penambahan sumber daya dan optimalisasi strategi, guna meningkatkan efisiensi dan efektivitas layanan darurat di Detroit merupakan hasil yang kita dapatkan. Selanjutnya, datta yang diberikan dalam format JSON dapat mempermudah kita untuk melakukan analisis lebih lanjut dan integrasi dengan sistem lain. Hal ini memungkinkan peningkatan berkelanjutan dalam layanan tanggap darurat.

Link Colab: https://colab.research.google.com/drive/1CltioUKcj_0swBdzIikaPkVIVdGjhlbr?usp=sharing

Daftar Pustaka

Fuchs, L. (2019). Analyzing 911 Call Data for Emergency Response Improvement. Journal of Emergency Management, 15(3), 123-135. Retrieved from https://www.jemjournal.com/article/S0019-0578(19)30202-3/fulltext