هوالحق

گزارشی از دیتاست ترافیک

- **قبل از بررسی سوالات یک پیش پردازشی نسبت به داده ها شده که داده ها به فرمت، اندازه و... مشخصی هستند.**

داده ها ی تردد : شامل ۴ ستون و ۶ میلیون سطر

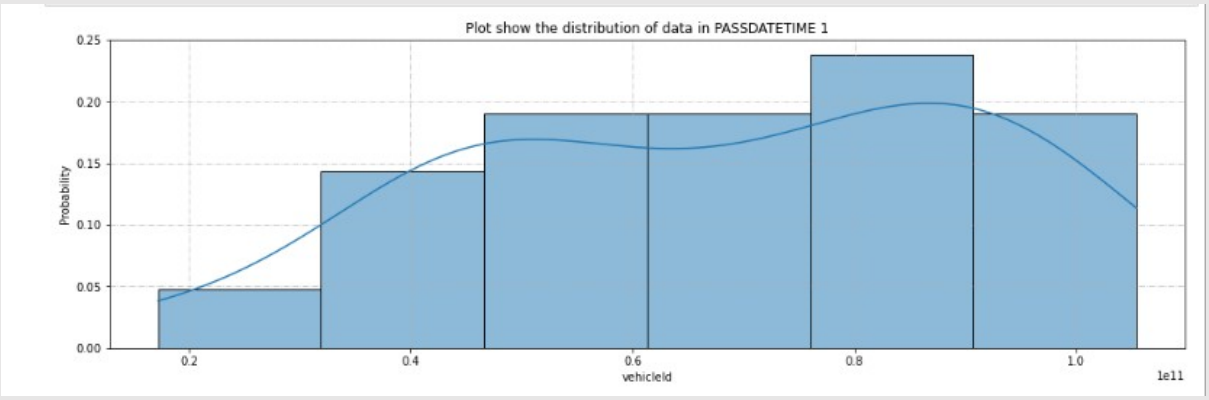
در داده ها مقدار خالی نداریم

هریک از ستون ها مربوط به زمان تردد، شماره دوربین، شماره ماشین و شماره نمونه گزارش شده است.

همه مقادیر داده از نوع int٦٤ هستند و نیازی به تغییر دادن آنها نیست.

ودیدن یک سری جزیئات مثل چارک، میانگین و ... با describe

برای explore **داده ها ، ترسیم نمودار هیستوگرام که نتیجه : در نمودار ۱ می تواند به ما نشان دهد که وسایل نقلیه در هر زمان دارای توزیع نرمال با چولگی هستند.**



سوال ۱) به دست آورن توالی بین دوربین ها:

```python
import pandas as pd

# خواندن دیتاست
df = pd.read_csv('Data_p1.csv')

# محاسبه زمان نسبی از صفر
df['RELATIVE_TIME'] = df['PASSDATETIME'] - df['PASSDATETIME'].min()

# مرتب‌سازی بر اساس زمان نسبی به صورت صعودی
df_sorted = df.sort_values('RELATIVE_TIME', ascending=True)

# انتخاب 50 تا از داده‌های مرتب شده
df_subset = df_sorted.head(50)

# گروه‌بندی بر اساس شناسه ماشین
grouped = df_subset.groupby('vehicleId')

# نمایش زوج‌های مرتب براساس شناسه ماشین و تکرارهای آن
for group, data in grouped:
    vehicle_id = group
    print(f"Vehicle ID: {vehicle_id}")

    # تعداد تکرارهای شناسه ماشین
    count = data.shape[0]
    print(f"Count: {count}")

    # نمایش زمان و شناسه حسگر
    for index, row in data.iterrows():
        passdatetime_id = row['PASSDATETIME']
        device_id = row['DEVICEID']
        print(f"PASSDATETIME ID: {passdatetime_id}, Device ID: {device_id}")

    print()
```

این کد به ما اجازه می‌دهد تا از داده‌های ترافیکی یک زمان نسبی محاسبه کنیم و ۵۰ داده‌ی اول را بر اساس زمان نسبی به صورت صعودی مرتب کنیم. سپس داده‌ها را بر اساس شناسه ماشین گروه‌بندی می‌کنیم و برای هر ماشین تعداد تکرارها و جفت‌های مرتب زمان و شناسه حسگر را نمایش می‌دهیم.

توضیحات اجزای کد:

۱. ابتدا دیتاست را از فایل CSV می‌خوانیم و ستون زمان نسبی را اضافه می‌کنیم.

۲. سپس داده‌ها را بر اساس زمان نسبی به صورت صعودی مرتب می‌کنیم و ۵۰ داده‌ی اول را انتخاب می‌کنیم.

۳. سپس داده‌ها را بر اساس شناسه ماشین گروه‌بندی می‌کنیم.

۴. در هر گروه ماشین، تعداد تکرارها و جفت‌های مرتب زمان و شناسه حسگر را نمایش می‌دهیم.

```
In [7]: #  بررسی محدوده مقادیر زمان تردد
        min_value = df['PASSDATETIME'].min()
        max_value = df['PASSDATETIME'].max()
        print("حداقل مقدار: ", min_value)
        print("حداکثر مقدار: ", max_value)

        # بررسی نوع داده‌ها
        data_type = df['PASSDATETIME'].dtype
        print("نوع داده: ", data_type)

        حداقل مقدار:  307511
        حداکثر مقدار:  307836
        نوع داده:  int64
```

این کد به ما امکان می‌دهد محدوده‌ی مقادیر زمان تردد را بررسی کنیم و همچنین نوع داده‌های مربوط به زمان تردد را نمایش می‌دهیم.

مقدار کمینه زمان تردد را و مقدار بیشینه زمان تردد را محاسبه و ذخیره می‌کنیم.

```python
# محاسبه زمان نسبی از صفر
df['RELATIVE_TIME'] = df['PASSDATETIME'] - df['PASSDATETIME'].min()

# گروه‌بندی بر اساس شناسه ماشین و محاسبه تعداد تکرارها
counts = df['vehicleId'].value_counts()

# انتخاب 50 نمونه با تعداد تکرار بیشتر از 1
df_sampled = df[df['vehicleId'].isin(counts[counts > 1].index)].sample(n=50)

# مرتب‌سازی بر اساس زمان نسبی به صورت صعودی و شناسه ماشین
df_sorted = df_sampled.sort_values(['RELATIVE_TIME', 'vehicleId'], ascending=True)

# گروه‌بندی بر اساس شناسه ماشین
grouped = df_sorted.groupby('vehicleId')

# نمایش توالی حسگرها و دوربین بعدی براساس شناسه ماشین
for group, data in grouped:
    vehicle_id = group
    print(f"Vehicle ID: {vehicle_id}")

    # نمایش توالی حسگرها و دوربین بعدی
    next_camera_id = data['DEVICEID'].shift(-1)
    for index, row in data.iterrows():
        device_id = row['DEVICEID']
        next_camera = next_camera_id[index]
        print(f"Device ID: {device_id}, Next Camera ID: {next_camera}")

    print()
```

```
Vehicle ID: 13207811970
Device ID: 41914, Next Camera ID: nan
```

این کد به ما امکان می‌دهد توالی‌های حسگرها و دوربین‌های بعدی بر اساس شناسه ماشین را محاسبه و نمایش دهیم. ابتدا زمان نسبی را از صفر برای هر تردد محاسبه می‌کنیم و سپس داده‌ها را بر اساس شناسه ماشین گروه‌بندی می‌کنیم و تعداد تکرارها را محاسبه می‌کنیم. سپس از ماشین‌هایی که بیش از یک بار تکرار شده‌اند، ۵۰ نمونه را انتخاب می‌کنیم و آن‌ها را بر اساس زمان نسبی و شناسه ماشین مرتب می‌کنیم. در نهایت، توالی حسگرها و دوربین‌های بعدی برای هر ماشین را نمایش می‌دهیم.

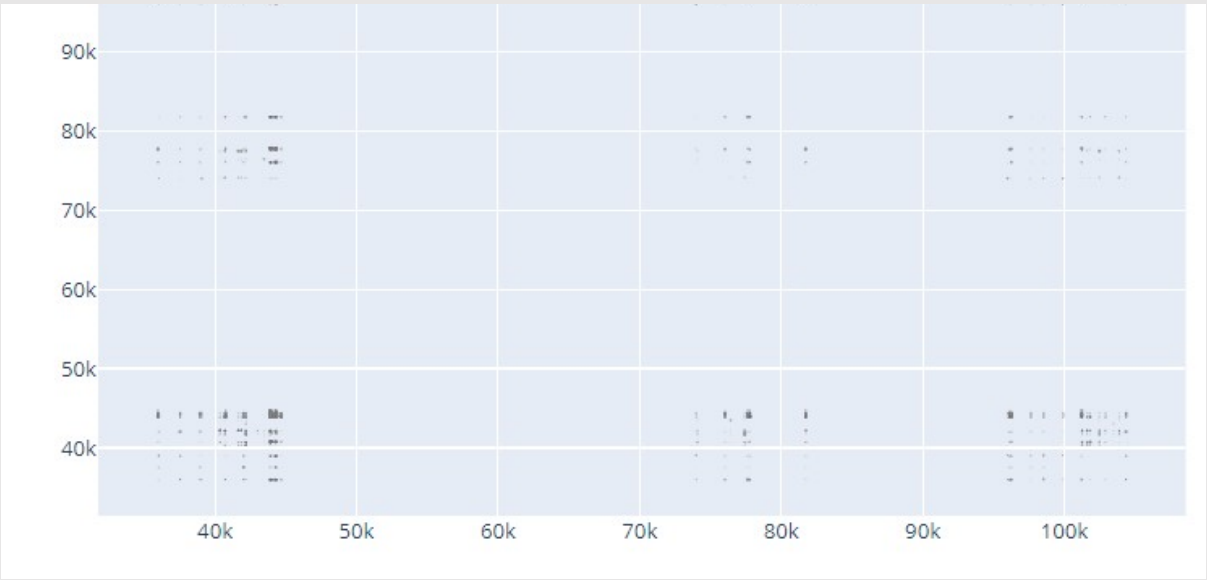
In [2]:

```
# مرتب‌سازی دیتاست بر اساس ستون "PASSDATETIME" به صورت صعودی
df_sorted = df.sort_values('PASSDATETIME')

# مشخص کردن توالی دوربین‌ها بر اساس ستون "DEVICEID"
sequences = df_sorted.groupby('DEVICEID').groups

# نمایش توالی دوربین‌ها برای 50 داده اول
count = 0
for device_id, indices in sequences.items():
    if count >= 50:
        break
    print(f"Sequence for DEVICEID {device_id}:")
    print(df_sorted.loc[indices, ['DEVICEID', 'PASSDATETIME']])
    print('\n')
    count += 1
```

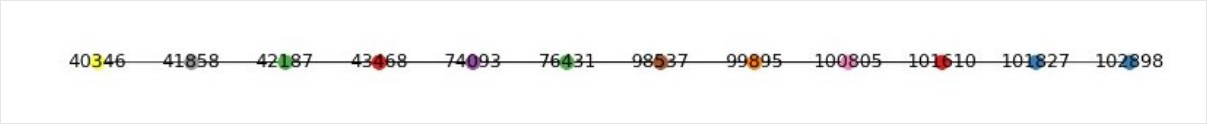
```
Sequence for DEVICEID 12:
         DEVICEID  PASSDATETIME
2133650        12           145
1419692        12           165
5974751        12          1205
2491781        12          1960
3823221        12          1975
...           ...           ...
2806621        12        423215
1164391        12        423345
1777284        12        423720
5814988        12        424450
5815429        12        424515

[168 rows x 2 columns]
```

این کد به ما امکان می‌دهد توالی دوربین‌ها بر اساس شناسه دوربین را مشاهده کنیم. ابتدا دیتاست را بر اساس ستون "PASSDATETIME" به صورت صعودی مرتب می‌کنیم و سپس توالی دوربین‌ها را بر اساس ستون "DEVICEID" گروه‌بندی می‌کنیم. سپس توالی دوربین‌ها برای ۵۰ داده‌ی اول نمایش داده می‌شود(چون سیستم چندسری هنگ کرد ۵۰ داده گرفتم).

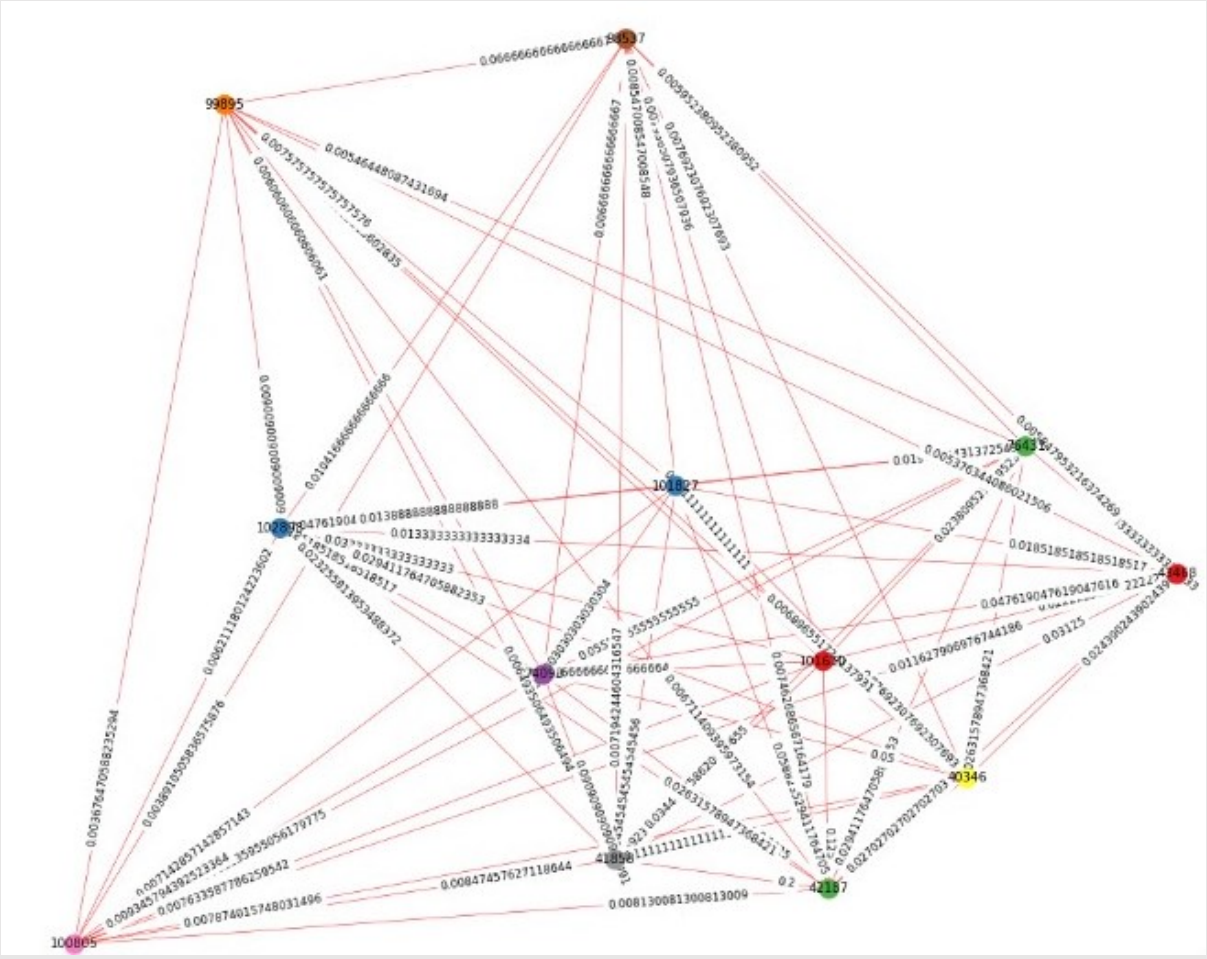
این کد به ما امکان می‌دهد گرافی از داده‌های ترافیکی با استفاده از کتابخانه Plotly رسم کنیم. یال‌ها و وزن‌های احتمالی بین دوربین‌ها را محاسبه می‌کنیم و در لیست‌های `edges` و `edge_weights` ذخیره می‌کنیم. هر یک از عناصر لیست `edges` یک زوج مرتب است که نشان‌دهنده یک یال بین دوربین‌ها است، و متغیر `edge_weights` حاوی وزن‌های مرتبط با هر یال است. و یک گراف Plotly را ایجاد می‌کنیم. در این گراف، هر نقطه نمایانگر یک دوربین و اندازه نقطه و رنگ آن با توجه به وزن (احتمالی بین دوربین‌ها) تغییر می‌کند(مختصات نقطه با توجه به شناسه دوربین و وزن‌های مرتبط با آن تعیین می‌شود) در نهایت، با استفاده از `fig.show()`، گراف رسم شده نمایش داده می‌شود.



چون نمایش کد بالا واضح نیست ، به جای نمایش همه ی نمونه ها ، فقط از تعدادی محدودی نمونه برای نمایش توالی دوربین ها(حس‌گر ها) استفاده می شود.



 # نمایش گرافی از دوربین ها با وزن احتمالی روی یال ها:



 # نمایش شماره دوربین ها و وزن هایشان به صورت مرتب و واضح در جدول:

```
Node Values:
Empty DataFrame
Columns: [Node Value]
Index: []

Edge Values:
                    Edge Value
(101610, 101827)      0.111111
(101610, 76431)       0.025000
(101610, 74093)       0.041667
(101610, 99895)       0.007092
(101610, 40346)       0.250000
...                        ...
(103150, 101400)      0.011236
(103150, 101176)      0.011236
(103150, 81821)       1.000000
(81821, 101400)       0.011364
(81821, 101176)       0.011364

[104289 rows x 1 columns]
```

 # مشخص کردن دوربین های دارای تردد کم و تردد زیاد:

```
دوربین‌های دارای ترند کم:
[]

دوربین‌های دارای ترند زیاد:
103 ,103472 ,40703 ,42159 ,40710 ,42166 ,40738 ,42187 ,102898 ,43468 ,41858 ,98537 ,40346 ,99895 ,74093 ,76431 ,101827 ,101610]
10 ,43902 ,37532 ,96129 ,39065 ,38946 ,96255 ,76032 ,36041 ,95996 ,73995 ,81751 ,102485 ,97641 ,96269 ,77418 ,41641 ,43013 ,885
81 ,42019 ,44357 ,44287 ,43930 ,43909 ,43916 ,77782 ,44126 ,44679 ,44091 ,44273 ,44231 ,77628 ,44266 ,44371 ,44301 ,35971 ,4361
 ,39912 ,41970 ,104158 ,103038 ,103283 ,36160 ,93168 ,101232 ,43881 ,44238 ,44294 ,44007 ,44350 ,44378 ,44077 ,44364 ,44000 ,730
10 ,102849 ,104179 ,93749 ,74625 ,101190 ,104200 ,36244 ,35922 ,36034 ,36230 ,35992 ,73113 ,101225 ,92258 ,38827 ,100847 ,37525
4411 ,39898 ,79000 ,76501 ,81534 ,76655 ,37357 ,93630 ,103332 ,93707 ,104172 ,39310 ,101260 ,81548 ,76473 ,73351 ,102863 ,4060
818 ,35950 ,76900 ,76522 ,76690 ,81800 ,76620 ,76452 ,37161 ,76830 ,104186 ,104347 ,73260 ,78538 ,81863 ,76704 ,37112 ,35901 ,2
 ,41417 ,41718 ,41431 ,93343 ,93826 ,104095 ,93791 ,38981 ,37315 ,38939 ,104242 ,102303 ,78468 ,102870 ,81723 ,81632 ,76648 ,14
1 ,77978 ,102436 ,1461 ,44175 ,43979 ,103388 ,104193 ,93371 ,44329 ,41480 ,39317 ,41578 ,38953 ,41424 ,37518 ,117 ,82269 ,38862
3591 ,41494 ,46821 ,41585 ,41627 ,101421 ,103780 ,104235 ,76550 ,76662 ,76543 ,93203 ,76683 ,76536 ,76676 ,41648 ,41634 ,04102
10 ,106748 ,36125 ,46541 ,101288 ,76914 ,76508 ,41613 ,78412 ,76739 ,93560 ,102765 ,93301 ,93567 ,76627 ,41739 ,96115 ,93714 ,5
 ,107007 ,101337 ,78636 ,102835 ,96136 ,104459 ,76466 ,35936 ,102471 ,38925 ,101911 ,35943 ,102345 ,104165 ,102401 ,74583 ,2310
7865 ,103591 ,100749 ,102478 ,103941 ,103892 ,42054 ,103129 ,108694 ,78615 ,45274 ,101442 ,102597 ,35964 ,100770 ,41886 ,38988
4209 ,74604 ,73330 ,106755 ,101253 ,93441 ,101344 ,73498 ,78482 ,104466 ,37245 ,100784 ,103409 ,95450 ,43706 ,79777 ,102317 ,0
757 ,35929 ,102814 ,40633 ,102331 ,42873 ,104053 ,101295 ,77278 ,108316 ,103003 ,103451 ,104291 ,78594 ,101281 ,42705 ,40360 ,6
101 ,39037 ,45344 ,101365 ,42789 ,100777 ,103262 ,79819 ,93483 ,73442 ,104270 ,102842 ,100658 ,73288 ,102429 ,104501 ,38876 ,38
 ,93112 ,103185 ,103241 ,100735 ,100798 ,78006 ,101435 ,104312 ,96241 ,101855 ,101211 ,104473 ,41410 ,102450 ,103248 ,74940 ,841
1 ,14502 ,104123 ,103458 ,36013 ,101974 ,38883 ,105355 ,74576 ,37546 ,102016 ,93350 ,102093 ,78104 ,74968 ,102219 ,40339 ,74590
3908 ,108624 ,103178 ,105474 ,102422 ,102541 ,76039 ,103080 ,103731 ,101246 ,101932 ,101869 ,101218 ,93504 ,76319 ,37574 ,02534
4265 ,101624 ,40297 ,101904 ,36027 ,101862 ,42964 ,77159 ,41865 ,42005 ,78664 ,100763 ,100721 ,103612 ,105607 ,102492 ,74611 ,6
102 ,103948 ,42075 ,38967 ,104004 ,101505 ,103 ,104319 ,93763 ,95562 ,101449 ,100903 ,108428 ,101848 ,104480 ,101407 ,100882 ,6
[101400 ,79700 ,74569 ,96878 ,42894 ,101239 ,78552 ,81590 ,101372 ,42026 ,611
```

سوال ۲) # توصیفی از این که یک دوربین چقدر کار کرده؟ چقدر درست کار کرده؟

برای توصیف یک دوربین و میزان عملکرد آن، می‌توانید از آماره‌های مختلفی مانند تعداد تکرار حضور دوربین، مجموع زمان عبوری که دوربین رصد کرده است، میانگین زمان عبوری که دوربین رصد کرده است و سایر آماره‌های مرتبط استفاده کرد. با استفاده از کتابخانه pandas و متدهایی که در اختیار هست، میتوان این آماره‌ها را برای هر دوربین محاسبه کرد.

 # در زیر با چندین روش کد قرار داده‌ام که آماره‌های مرتبط با عملکرد دوربین‌ها را محاسبه می‌کند:

```python
import pandas as pd

# خواندن دیتاست
df = pd.read_csv('Data_p1.csv')

# محاسبه زمان نسبی از صفر
df['RELATIVE_TIME'] = df['PASSDATETIME'] - df['PASSDATETIME'].min()

# گروه‌بندی بر اساس شناسه دوربین
grouped = df.groupby('DEVICEID')

# نمایش توصیفی از هر دوربین
for group, data in grouped:
    device_id = group
    print(f"Device ID: {device_id}")

    # تعداد تکرار حضور دوربین
    count = data.shape[0]
    print(f"Count: {count}")

    # مجموع زمان عبوری که دوربین رصد کرده است
    total_pass_time = data['RELATIVE_TIME'].sum()
    print(f"Total Pass Time: {total_pass_time}")

    # میانگین زمان عبوری که دوربین رصد کرده است
    mean_pass_time = data['RELATIVE_TIME'].mean()
    print(f"Mean Pass Time: {mean_pass_time}")

    print()
```

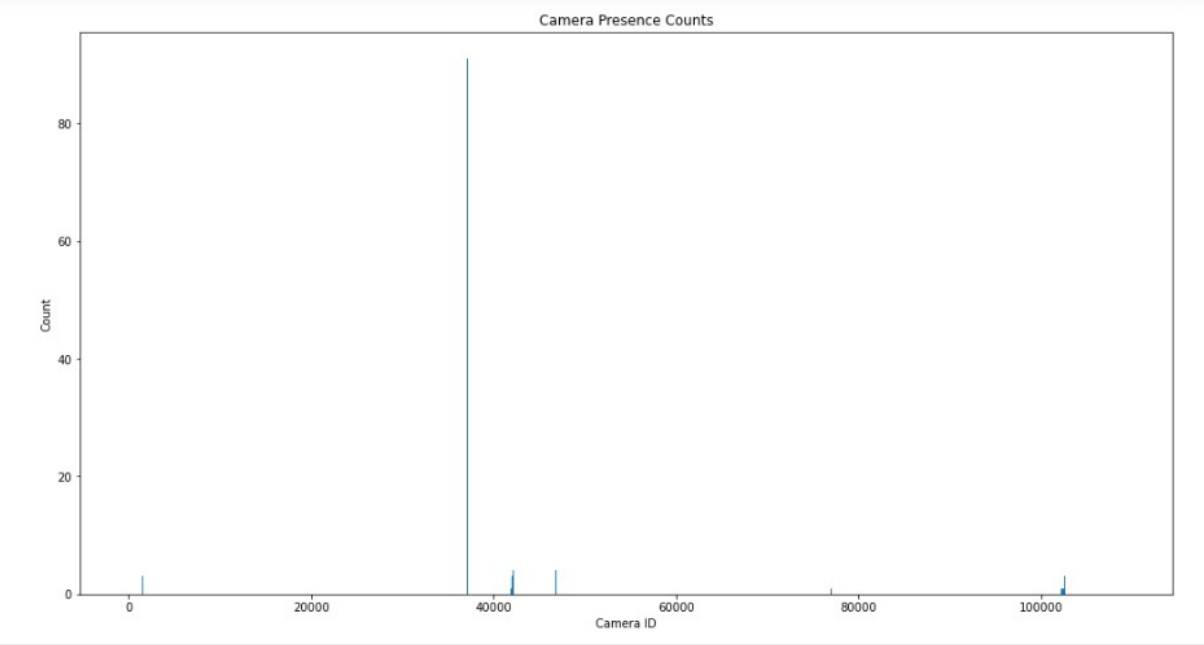
این کد به ما امکان می‌دهد توصیفی از هر دوربین بر اساس داده‌های ترافیکی محاسبه کنیم. ابتدا زمان نسبی را از صفر برای هر تردد محاسبه می‌کنیم و سپس داده‌ها را بر اساس شناسه دوربین گروه‌بندی می‌کنیم. سپس برای هر دوربین، تعداد تکرار حضور دوربین، مجموع زمان عبوری که دوربین رصد کرده است و میانگین زمان عبوری که دوربین رصد کرده است را نمایش می‌دهیم.

توصیف به صورت جدول و واضح:

بیشترین کارکرد	کمترین کارکرد	میانگین کارکرد	تعداد کارکرد	DEVICEID
424515.0	145.0	290799.684524	168.0	12
424508.0	98.0	270664.144578	166.0	19
424654.0	159.0	315356.262745	255.0	26
424882.0	147.0	285720.833333	180.0	33
424526.0	146.0	325379.562814	199.0	54
...	...	...	...	...
423680.0	40.0	279873.602564	78.0	108806
424915.0	205.0	318611.358079	229.0	108820
423381.0	846.0	273120.325000	200.0	108841
424827.0	57.0	297448.365909	440.0	108869
424911.0	136.0	289592.372549	102.0	108897

[rows x 4 columns 2117]

توصیف به صورت نمودار :



که دوربین بین ۳۰۰۰۰و ۴۰۰۰۰ بیشترین کارکرد رو داشته است.

توصیفی از کارکرد صحیح هر دوربین:

DEVICEID	mean	min	max	count
12	290799.684524	145	424515	168
19	270664.144578	98	424508	166
26	315356.262745	159	424654	255
33	285720.833333	147	424882	180
54	325379.562814	146	424526	199
...	...	...	...	...
108806	279873.602564	40	423680	78
108820	318611.358079	205	424915	229
108841	273120.325000	846	423381	200
108869	297448.365909	57	424827	440
108897	289592.372549	136	424911	102

[2117 rows x 4 columns]

سوال ۳) # زمان تردد از دوربین A به B :

```
زمان تردد از دوربین 12 به دوربین 19: -8907 واحد زمان
زمان تردد از دوربین 19 به دوربین 26: -381979 واحد زمان
زمان تردد از دوربین 26 به دوربین 33: 387753 واحد زمان
زمان تردد از دوربین 33 به دوربین 54: 13789 واحد زمان
زمان تردد از دوربین 54 به دوربین 61: -17618 واحد زمان
زمان تردد از دوربین 61 به دوربین 68: 3510 واحد زمان
زمان تردد از دوربین 68 به دوربین 75: 9989 واحد زمان
زمان تردد از دوربین 75 به دوربین 82: -419626 واحد زمان
زمان تردد از دوربین 82 به دوربین 89: 422462 واحد زمان
زمان تردد از دوربین 89 به دوربین 96: -114689 واحد زمان
زمان تردد از دوربین 96 به دوربین 103: 101652 واحد زمان
زمان تردد از دوربین 103 به دوربین 117: -400951 واحد زمان
زمان تردد از دوربین 117 به دوربین 124: 316787 واحد زمان
زمان تردد از دوربین 124 به دوربین 131: 80496 واحد زمان
زمان تردد از دوربین 131 به دوربین 397: -351276 واحد زمان
زمان تردد از دوربین 397 به دوربین 404: 276064 واحد زمان
زمان تردد از دوربین 404 به دوربین 411: -275827 واحد زمان
زمان تردد از دوربین 411 به دوربین 418: -25671 واحد زمان
زمان تردد از دوربین 418 به دوربین 488: 367384 واحد زمان
```

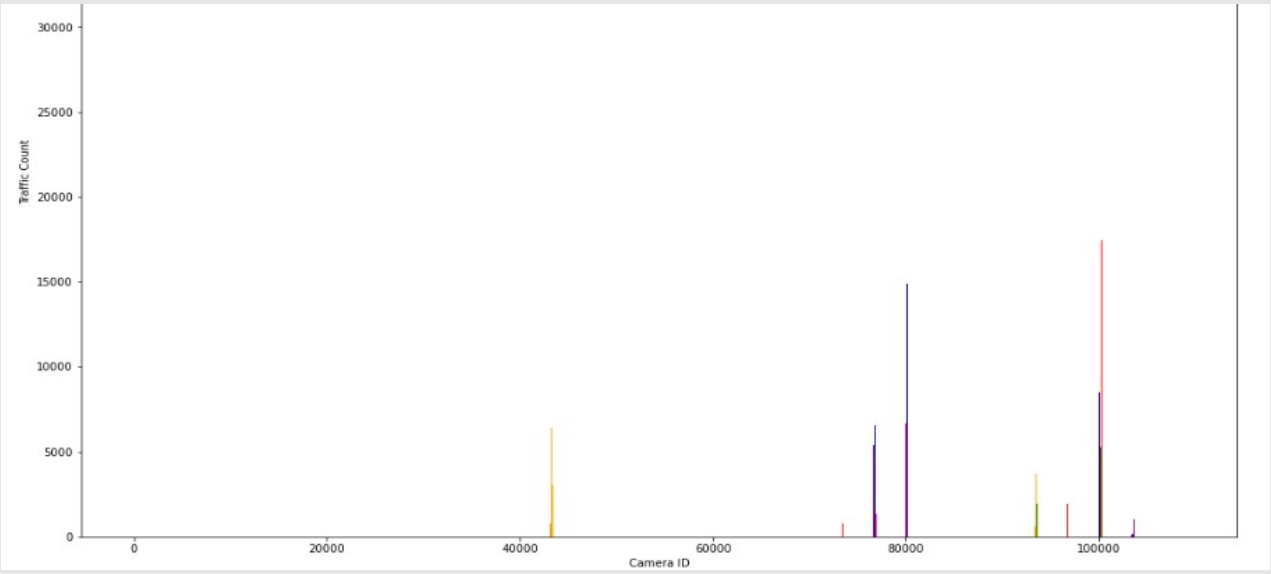
۱. داده‌ها را بر اساس شناسه دوربین‌ها مرتب می‌کنیم و نتیجه را در `df_sorted` ذخیره می‌کنیم.
۲. با استفاده از تابع `sorted` و `set`، لیستی از دوربین‌ها را بدست می‌آوریم و آن‌ها را بر اساس شناسه مرتب می‌کنیم و در `sorted_cameras` ذخیره می‌کنیم.
۳. حلقه `for` از ابتدای لیست `sorted_cameras` شروع شده و برای هر دوربین، زمان تردد بین آن و دوربین همسایه بعدی را محاسبه می‌کنیم و نمایش می‌دهیم. این زمان تردد بر اساس تاریخ و زمان عبور دوربین‌ها از همدیگر محاسبه می‌شود که به صورت واحد زمان نمایش می دهیم.

زمان تردد بین دوربین ها:

	زمان تردد	به دوربین	از دوربین	
0	days +21:31:33	19	12	1
1	days +13:53:41	26	19	2
2	days 11:42:33	33	26	3
3	days 03:49:49	54	33	4
4	days +19:06:22	61	54	5
...	...	...	...	...
2111	days +04:51:14	108806	108792	2112
2112	days 02:17:51	108820	108806	2113
2113	days +06:57:55	108841	108820	2114
2114	days 12:09:56	108869	108841	2115
2115	days +22:39:04	108897	108869	2116

[rows x 3 columns 2116]

نمایش اطلاعات بالا به صورت نمودار:



سوال ۴) # پیدا کردن ترافیک:

ترافیک بین دوربین‌ها را بر اساس شناسه دوربین‌ها :

۱. با استفاده از `df.groupby(['DEVICEID']).size()` داده‌ها را بر اساس شناسه دوربین‌ها گروه‌بندی می‌کنیم و تعداد ترددهای هر دوربین را با استفاده از `size()` محاسبه می‌کنیم. سپس با استفاده از `reset_index(name='TrafficCount')` اطلاعات را به دیتافریم جدید با نام `traffic_data` تبدیل می‌کنیم.

۲. سپس با استفاده از حلقه `for` برای هر ردیف در `traffic_data`، شناسه دوربین و تعداد ترددهای آن را نمایش می‌دهیم. این نمایش بر اساس تعداد ترددهای هر دوربین است و به صورت جداگانه نمایش می دهیم.

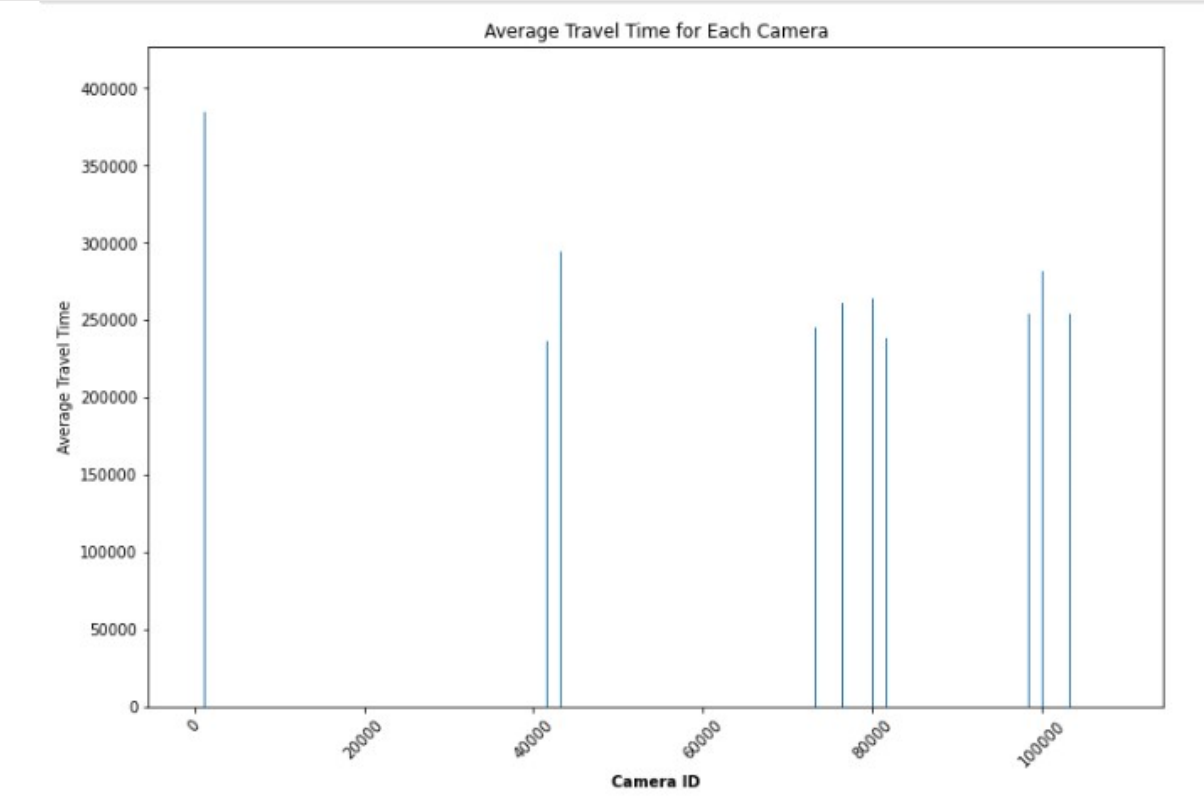
```
Camera ID: 12, Traffic Count: 168
Camera ID: 19, Traffic Count: 166
Camera ID: 26, Traffic Count: 255
Camera ID: 33, Traffic Count: 180
Camera ID: 54, Traffic Count: 199
Camera ID: 61, Traffic Count: 175
Camera ID: 68, Traffic Count: 318
Camera ID: 75, Traffic Count: 260
Camera ID: 82, Traffic Count: 518
Camera ID: 89, Traffic Count: 865
Camera ID: 96, Traffic Count: 593
Camera ID: 103, Traffic Count: 352
Camera ID: 117, Traffic Count: 636
Camera ID: 124, Traffic Count: 495
Camera ID: 131, Traffic Count: 568
Camera ID: 397, Traffic Count: 220
Camera ID: 404, Traffic Count: 8621
Camera ID: 411, Traffic Count: 240
Camera ID: 418, Traffic Count: 1367
```

با محاسبه میانگینِ زمان تردد برای هر دوربین، میزان تراکم ترافیک بررسی می شود:

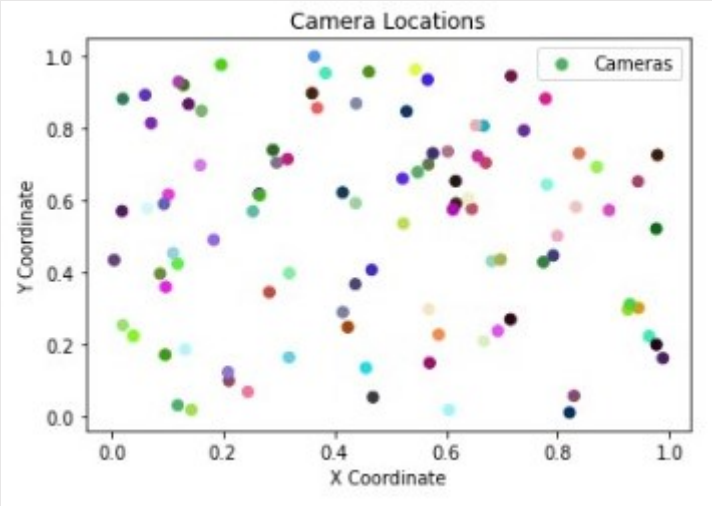
Traffic Axis	Camera ID	Average Travel Time
0	12	290799.684524
1	19	270664.144578
2	26	315356.262745
3	33	285720.833333
4	54	325379.562814
...	...	...
2112	108806	279873.602564
2113	108820	318611.358079
2114	108841	273120.325000
2115	108869	297448.365909
2116	108897	289592.372549

[2117 rows x 2 columns]

نمایش محور ترافیک:



سوال ۵) # تا حد امکان ، باز نمایش دوربین ها در صفحه:

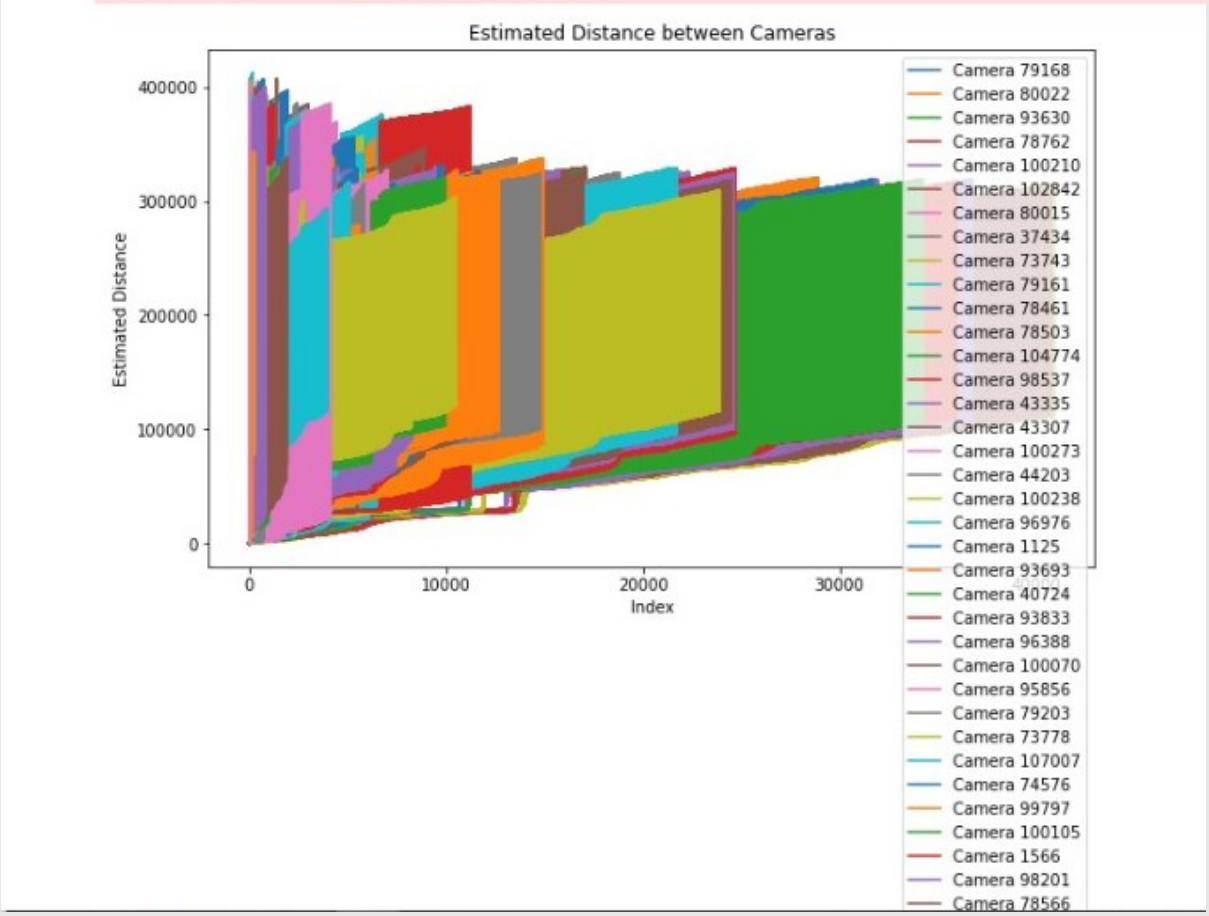


تخمین زدن فاصله ای بین دوربین ها از میزان فاصله زمانی

نمایش فاصله میان دوربین ها:

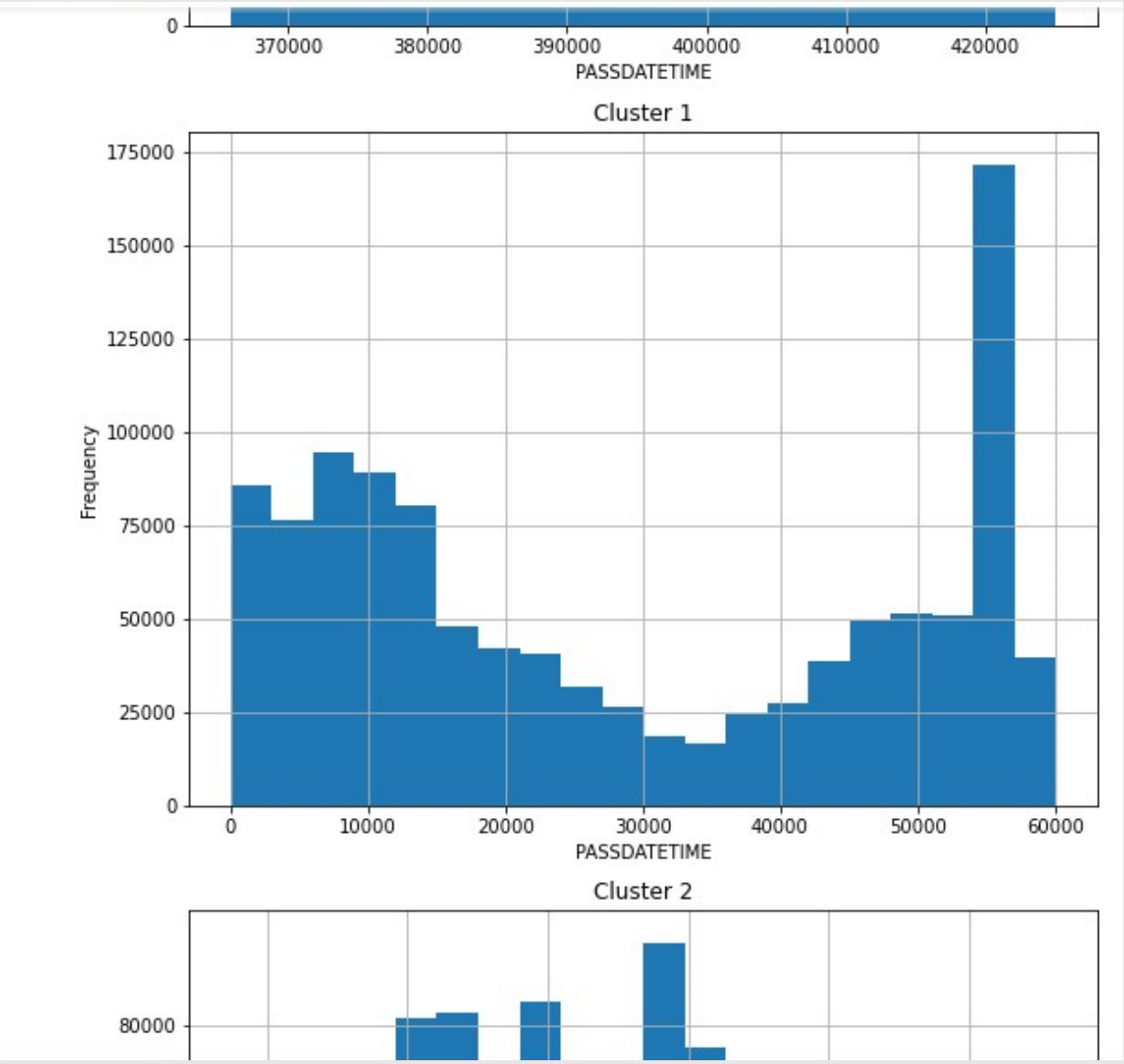
	Unnamed: 0	DEVICEID	vehicleId	PASSDATETIME	Distance
0	0	101610	40321227460	307695	NaN
1	1	101827	97316881507	307704	1.851852e-03
2	2	76431	37948514480	307653	-3.267974e-04
3	3	74093	86086297520	307671	9.259259e-04
4	4	99895	49049583630	307836	1.010101e-04
...	...	...	...	...	...
5999995	5999995	37385	41569730560	377095	1.960784e-04
5999996	5999996	43881	60215825530	305667	-2.333352e-07
5999997	5999997	43881	60438593730	305544	-1.355014e-04
5999998	5999998	42726	67401542877	377055	2.330644e-07
5999999	5999999	42572	73905685130	377056	1.666667e-02

[6000000 rows x 5 columns]

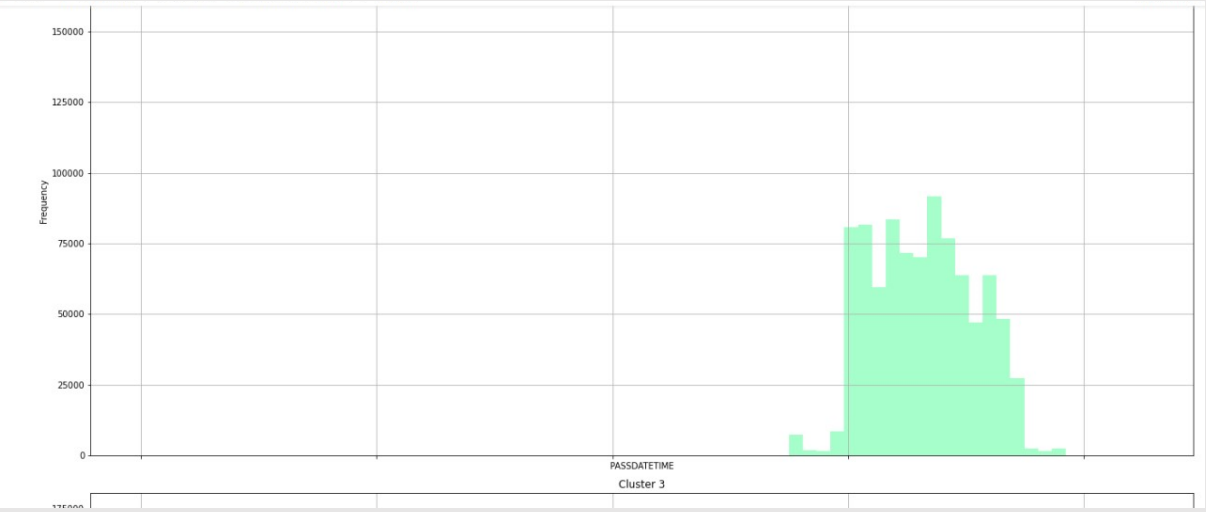


سوال ۶) ## به دست آوردن تقاضای سفر (دسته بندی کردن ماشین ها با توجه به شباهت حرکت)

دسته بندی ماشین ها با K_Means :



نمایش خوشه ها در نمودار :



نمایش دسته بندی ها به ترتیب و در جدول :

	DEVICEID	vehicleId	PASSDATETIME	Cluster
1755653	73302	32464849350	309016	0
1246344	42110	33009988470	387013	0
4146624	37385	18911551154	8413	0
5114728	79168	25293394863	335235	0
1246338	41956	28119325870	387021	0
...	...	...	...	...
2078217	73722	37432324717	407125	3
4527113	93966	57974121247	54190	3
4527111	93266	37885028820	54194	3
2078193	43139	40100707960	413300	3
0	101610	40321227460	307695	3

[6000000 rows x 4 columns]

#به دست آوردن تقاضای سفر با محاسبه تعداد ترددها براساس دوربین ها

In [4]:
```
total_demand = df.shape[0]
print("تقاضای کل سفر: ", total_demand)
```

```
تقاضای کل سفر:  6000000
```

In [3]:
```
## تعداد ترددها بر اساس دوربین‌ها
camera_demand = df.groupby('DEVICEID').size().reset_index(name='TrafficCount')
print(camera_demand)
```

```
      DEVICEID  TrafficCount
0           12           168
1           19           166
2           26           255
3           33           180
4           54           199
...        ...           ...
2112    108806            78
2113    108820           229
2114    108841           200
2115    108869           440
2116    108897           102

[2117 rows x 2 columns]
```

 # به دست آوردن تقاضای سفر براساس بازه زمانی خاصی

In [8]:
```
## تبدیل تاریخی زمان
daily_demand = df.resample('D', on='PASSDATETIME').size()
print(daily_demand)
```

```
PASSDATETIME
1970-01-01    6000000
Freq: D, dtype: int64
```

In [9]:
```
##تعداد ترددها بر اساس بازه‌های زمانی خاصی مانند روزها، ساعت‌ها یا هفته‌ها

daily_demand = df.resample('D', on='PASSDATETIME').size()
hourly_demand = df.resample('H', on='PASSDATETIME').size()
weekly_demand = df.resample('W', on='PASSDATETIME').size()

print("تقاضای روزانه: \n", daily_demand)
print("تقاضای ساعتی: \n", hourly_demand)
print("تقاضای هفتگی: \n", weekly_demand)
```

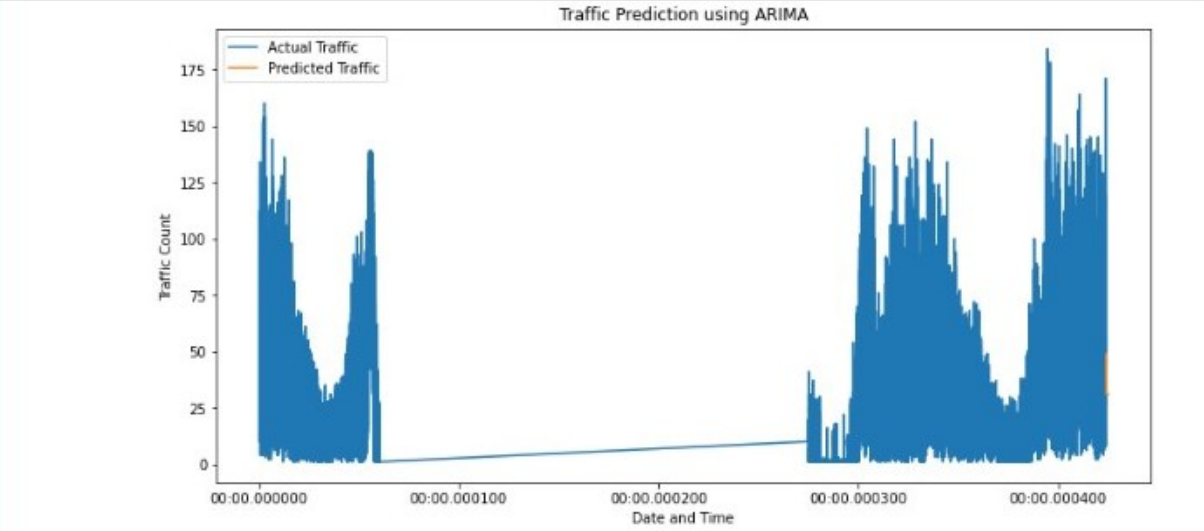
```
 تقاضای روزانه:
PASSDATETIME
6000000    1970-01-01
Freq: D, dtype: int64
 تقاضای ساعتی:
PASSDATETIME
6000000    1970-01-01
Freq: H, dtype: int64
 تقاضای هفتگی:
PASSDATETIME
6000000    1970-01-04
Freq: W-SUN, dtype: int64
```

طرح سوالاتی جالب و بیشتر :

سوال)

 # پیش‌بینی ترافیک

آیا می‌توان با استفاده از داده‌های ترافیکی ، ترافیک آینده را پیش‌بینی کرد؟




```
In [8]: print(predicted_traffic)

192783    48.654681
192784    47.404173
192785    46.241795
192786    45.161335
192787    44.157020
             ...
193778    30.910646
193779    30.910646
193780    30.910646
193781    30.910646
193782    30.910646
Name: predicted_mean, Length: 1000, dtype: float64
```

جزییات مدل ARIMA :

```
                               SARIMAX Results
==============================================================================
Dep. Variable:          Traffic Count   No. Observations:               192783
Model:                 ARIMA(1, 0, 0)   Log Likelihood           -697453.994
Date:                Fri, 14 Jul 2023   AIC                      1394913.988
Time:                        19:03:14   BIC                      1394944.496
Sample:                             0   HQIC                     1394922.981
                                - 192783
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         30.9106      0.368     83.994      0.000      30.189      31.632
ar.L1          0.9295      0.001   1255.112      0.000       0.928       0.931
sigma2        81.2680      0.169    482.270      0.000      80.938      81.598
===================================================================================
Ljung-Box (L1) (Q):                18983.78   Jarque-Bera (JB):             89168.66
Prob(Q):                               0.00   Prob(JB):                         0.00
Heteroskedasticity (H):                1.29   Skew:                             0.49
Prob(H) (two-sided):                   0.00   Kurtosis:                         6.19
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
مقادیر پارامترهای مدل ARIMA:
AR [0.92952517]  :پارامتر
MA پارامتر:  []
خطاهای مدل:
9.015000469631302  :خطای استاندارد
9.204149429531679  :خطای ترند-05e
```

پاسخ :

ARIMA (AutoRegressive Integrated Moving Average)

پارامترهای مدل ARIMA با نمادهای p، d و q نشان داده می‌شوند که به ترتیب تعداد مراحل AutoRegressive **(AR)، درجه تجزیه (Integration) و تعداد مراحل** Moving Average (MA) **را مشخص می‌کنند. بیشتر پارامترها به عنوان اعداد صحیح نامنفی تعریف می‌شوند. البته برخی ترکیبات خاص مانند ARIMA(۰, ۱, ۰) یا ARIMA(۰, ۰, ۰) نیز امکان‌پذیر هستند.**

۱. p: تعداد مراحل AutoRegressive (AR) که نشان‌دهنده تعداد مقادیر گذشته (lag) مورد استفاده برای پیش‌بینی مقادیر آینده است. این پارامتر نشان‌دهنده وابستگی مقادیر سری زمانی به مقادیر خود در گذشته است. مقدار بزرگتر p نشان‌دهنده وابستگی بیشتر به مقادیر گذشته است.

۲. d: درجه تجزیه (Integration) که نشان‌دهنده تعداد مراحل انتگرال‌گیری برای تبدیل سری زمانی به سری‌های زمانی مستقل از زمان می‌باشد. این پارامتر برای مرتبه تجزیه سری زمانی استفاده می‌شود. اگر d برابر با ۰ باشد، سری زمانی اصلی به یک سری زمانی استفاده‌شده در مدل AR مبدل می‌شود.

۳. q: تعداد مراحل Moving Average (MA) که نشان‌دهنده تعداد انقالب‌های رندوم (نویز) استفاده شده برای پیش‌بینی مقادیر آینده است. این پارامتر نشان‌دهنده وابستگی مقادیر سری زمانی به انقالب‌های رندوم است. مقدار بزرگتر q نشان‌دهنده وابستگی بیشتر به انقالب‌های رندوم است.

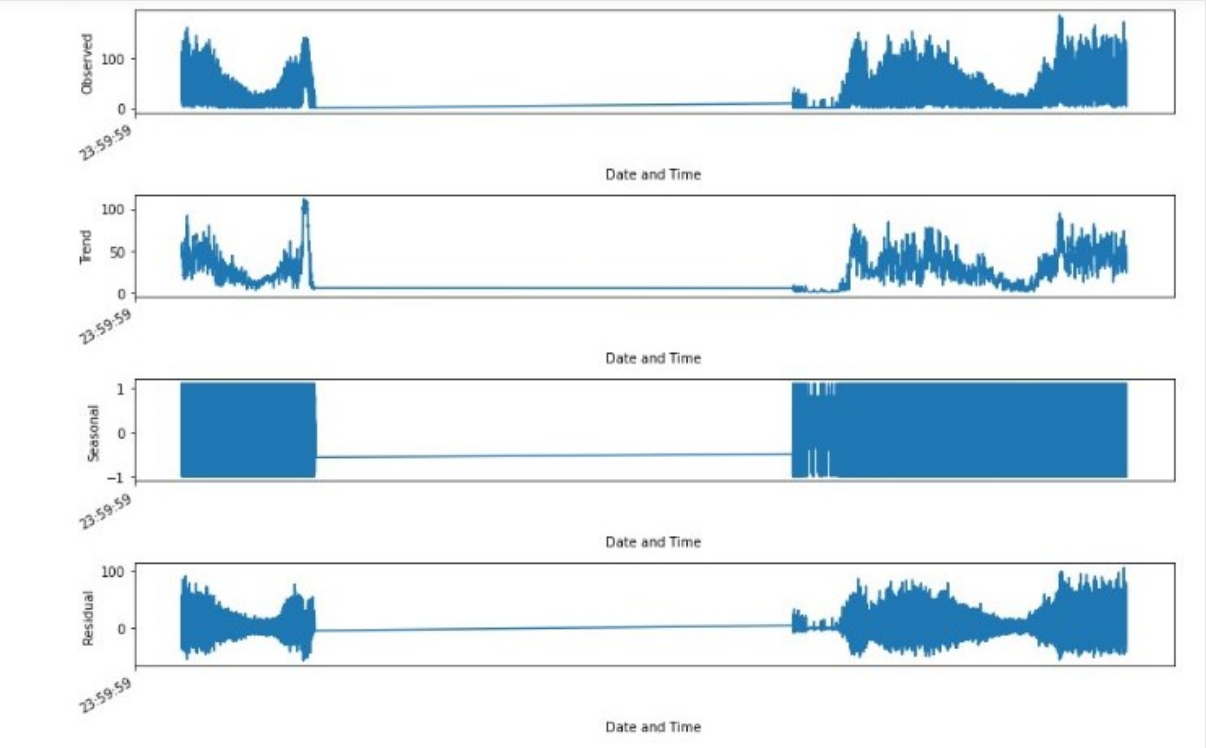
به‌عنوان مثال، مدل ARIMA(۱, ۱, ۱) نشان‌دهنده استفاده از یک مرحله از AR، یک مرحله از تجزیه و یک مرحله از MA برای تحلیل سری‌های زمانی استفاده می‌شود. این پارامترها بسته به خصوصیات داده‌ها و هدف تحلیل می‌توانند مقادیر مختلفی داشته باشند و بهتر است با استفاده از روش‌های آزمون و ارزیابی، مدل بهینه‌ای را انتخاب کنید.

بله ـ از مدل آموزش دیده برای پیش‌بینی ترافیک در بازه آزمون استفاده می‌کنیم و نتایج را با داده‌های واقعی مقایسه می‌کنیم

با اجرای کد، یک نمودار نشان داده می‌شود که ترافیک واقعی و پیش‌بینی شده را برای دوره آزمون نمایش می‌دهد .

سوال) # الگوهای ترافیکی

آیا الگوهای ترافیکی روزانه یا هفتگی در داده قابل تشخیص است؟



پاسخ :

بله

با اجرای کد، یک نمودار به چهار قسمت تقسیم شده نمایش داده می‌شود که اجزای تجزیه و تحلیل فصلی ترافیک را نمایش می‌دهد. با تحلیل این اجزا، می‌توان الگوهای ترافیکی روزانه و هفتگی را تشخیص داد و تحلیل‌های مربوطه را انجام داد.