JSON FILE:-

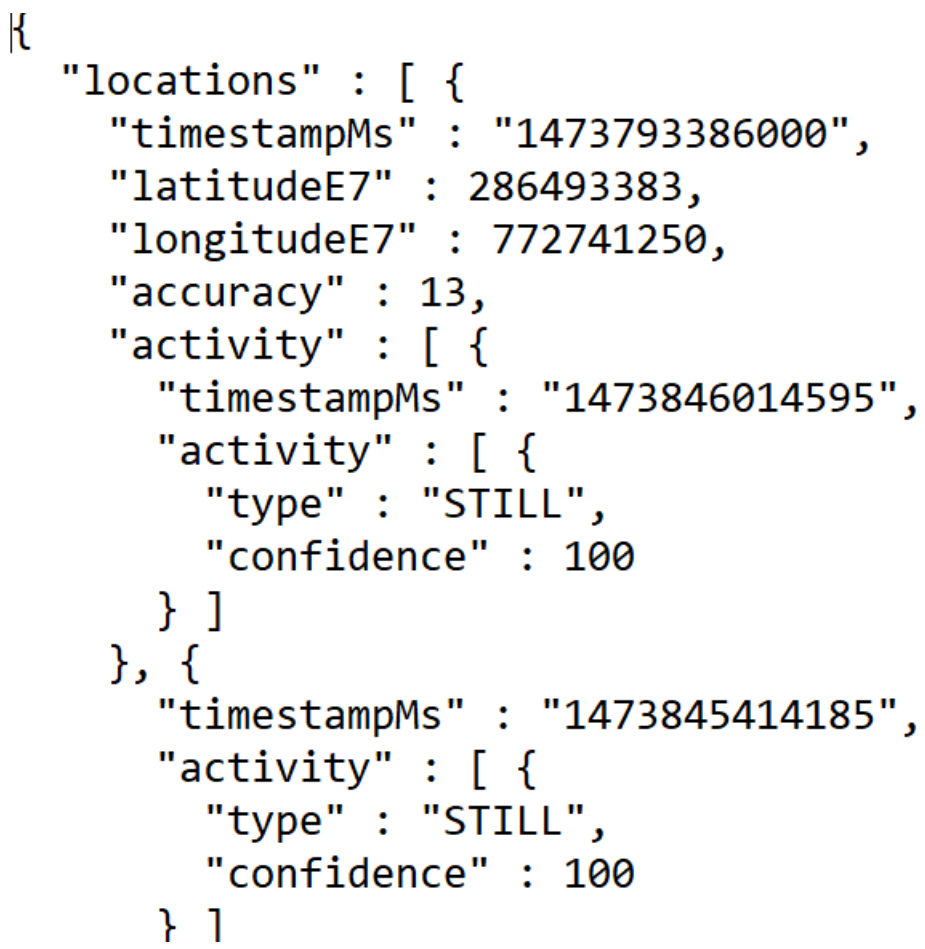
To get the data of location history, the JSON file is downloaded from google takeout.

- A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format.
- It is primarily used for transmitting data between a web application and a server

In [61]:

```
from IPython.display import Image
Image(filename='json.png')
```

Out[61]:



In [46]:

```
#using
import pandas as pd
from datetime import datetime as dt
# load the google location history data
df_gps = pd.read_json('location.json')
print('There are {:,} rows in the location history dataset'.format(len(df_gps)))
```

```
There are 322,903 rows in the location history dataset
```

In [47]:

```
# EXTRACTING LATITUDE,LONGITUDE AND TIMSTAMP_MS
df_gps['lat'] = df_gps['locations'].map(lambda x: x['latitudeE7'])
df_gps['lon'] = df_gps['locations'].map(lambda x: x['longitudeE7'])
df_gps['timestamp_ms'] = df_gps['locations'].map(lambda x: x['timestampMs'])

# convert lat/lon to decimalized degrees and the timestamp to date-time
```

```
# convert lat/lon to decimalized degrees and the timestamp to date-time

df_gps['Latitude'] = df_gps['lat'] / 10.**7
df_gps['Longitude'] = df_gps['lon'] / 10.**7

#timestamps are in ISO 8601 standard

df_gps['timestamp_ms'] = df_gps['timestamp_ms'].astype(float) / 1000

#converting timestamps to datetime format

df_gps['datetime'] = df_gps['timestamp_ms'].map(lambda x: dt.fromtimestamp(x).strftime('%Y-%m-%d %H
:%M:%S'))
#date_range = '{}-{}'.format(df_gps['datetime'].min()[:4], df_gps['datetime'].max()[:4])
```

In [48]:

```
df_gps.head()
```

Out[48]:

	locations	lat	lon	timestamp_ms	Latitude	Longitude	datetime
0	{'timestampMs': '1473793386000', 'latitudeE7':...	286493383	772741250	1.473793e+09	28.649338	77.274125	2016-09-14 00:33:06
1	{'timestampMs': '1473846266000', 'latitudeE7':...	286485800	772737317	1.473846e+09	28.648580	77.273732	2016-09-14 15:14:26
2	{'timestampMs': '1473905928000', 'latitudeE7':...	286490317	772746167	1.473906e+09	28.649032	77.274617	2016-09-15 07:48:48
3	{'timestampMs': '1473990648000', 'latitudeE7':...	286478617	772745883	1.473991e+09	28.647862	77.274588	2016-09-16 07:20:48
4	{'timestampMs': '1474095329000', 'latitudeE7':...	286489033	772741467	1.474095e+09	28.648903	77.274147	2016-09-17 12:25:29

In [50]:

```
# drop columns we don't need, then show a slice of the dataframe
df_gps = df_gps.drop(labels=['locations', 'timestamp_ms'], axis=1, inplace=False)
df_gps.head()
```

Out[50]:

	lat	lon	Latitude	Longitude	datetime
0	286493383	772741250	28.649338	77.274125	2016-09-14 00:33:06
1	286485800	772737317	28.648580	77.273732	2016-09-14 15:14:26
2	286490317	772746167	28.649032	77.274617	2016-09-15 07:48:48
3	286478617	772745883	28.647862	77.274588	2016-09-16 07:20:48
4	286489033	772741467	28.648903	77.274147	2016-09-17 12:25:29

Finding address using geopy library

- # Reverse geocoding:-It starts with a latitude & longitude point, and gives us back the corresponding street address.
- Geopy library is used for this purpose.
- Geopy is a Python 2 and 3 client for popular geocoding web services.
- Geopy is used to locate the coordinates of addresses, cities, countries, and landmarks across the globe.

In [51]:

```
import geopy
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter

subdf = df_gps.head(10).copy(deep=True)
#creating tuple of latitude and longitude
subdf["LatLong"] = subdf["Latitude"].map(str) + ',' + subdf["Longitude"].map(str)
```

```
subdf.head()
```

Out[51]:

	lat	lon	Latitude	Longitude	datetime	LatLong
0	286493383	772741250	28.649338	77.274125	2016-09-14 00:33:06	28.6493383,77.274125
1	286485800	772737317	28.648580	77.273732	2016-09-14 15:14:26	28.64858,77.2737317
2	286490317	772746167	28.649032	77.274617	2016-09-15 07:48:48	28.6490317,77.2746167
3	286478617	772745883	28.647862	77.274588	2016-09-16 07:20:48	28.6478617,77.2745883
4	286489033	772741467	28.648903	77.274147	2016-09-17 12:25:29	28.6489033,77.2741467

Nominatim geocoder

- Nominatim uses OpenStreetMap data to find an address for any location on the planet.
- User_Agent is an http request header that is sent with each request.
- Nominatim requires this value to be set to our application name.
- The goal is to be able to limit the number of requests per application.

In [52]:

```
locator = Nominatim(user_agent="myGeocoder", timeout=10)
rgeocode = RateLimiter(locator.reverse, min_delay_seconds=0.001)
subdf['address'] = subdf['LatLong'].apply(rgeocode)

#  detailed addresses will be shown
subdf.head(10)
```

Out[52]:

	lat	lon	Latitude	Longitude	datetime	LatLong	address
0	286493383	772741250	28.649338	77.274125	2016-09-14 00:33:06	28.6493383,77.274125	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
1	286485800	772737317	28.648580	77.273732	2016-09-14 15:14:26	28.64858,77.2737317	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
2	286490317	772746167	28.649032	77.274617	2016-09-15 07:48:48	28.6490317,77.2746167	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
3	286478617	772745883	28.647862	77.274588	2016-09-16 07:20:48	28.6478617,77.2745883	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
4	286489033	772741467	28.648903	77.274147	2016-09-17 12:25:29	28.6489033,77.2741467	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
5	286390117	772658299	28.639012	77.265830	2016-09-17 18:08:54	28.6390117,77.2658299	(Geeta Colony Road, Gandhi Nagar Tehsil, East ...
6	286487450	772741817	28.648745	77.274182	2016-09-18 03:40:36	28.648745,77.2741817	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
7	286488367	772743383	28.648837	77.274338	2016-09-18 08:06:16	28.6488367,77.2743383	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
8	286488267	772743317	28.648827	77.274332	2016-09-18 11:20:03	28.6488267,77.2743317	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...
9	286487128	772744513	28.648713	77.274451	2016-09-18 11:20:31	28.6487128,77.2744513	(Geeta Colony, Preet Vihar Tehsil, East Delhi,...

In [54]:

```
latlong_col = df_gps[['Latitude','Longitude']]
print(latlong_col)
l1=latlong_col.head(20)
print(l1)
```

```
    Latitude  Longitude
0  28.649338  77.274125
1  28.648580  77.273732
2  28.649032  77.274617
3  28.647862  77.274588
```

```
4       28.648903  77.274147
...           ...        ...
322898  28.650049  77.273186
322899  28.650049  77.273186
322900  28.650049  77.273186
322901  28.650279  77.272303
322902  28.650279  77.272303

[322903 rows x 2 columns]
     Latitude  Longitude
0   28.649338  77.274125
1   28.648580  77.273732
2   28.649032  77.274617
3   28.647862  77.274588
4   28.648903  77.274147
5   28.639012  77.265830
6   28.648745  77.274182
7   28.648837  77.274338
8   28.648827  77.274332
9   28.648713  77.274451
10  28.648690  77.274397
11  28.648735  77.274268
12  28.647053  77.272303
13  28.648810  77.274419
14  28.648706  77.274228
15  28.646763  77.273021
16  28.648943  77.274693
17  28.648943  77.274693
18  28.647053  77.272303
19  28.648943  77.274693
```

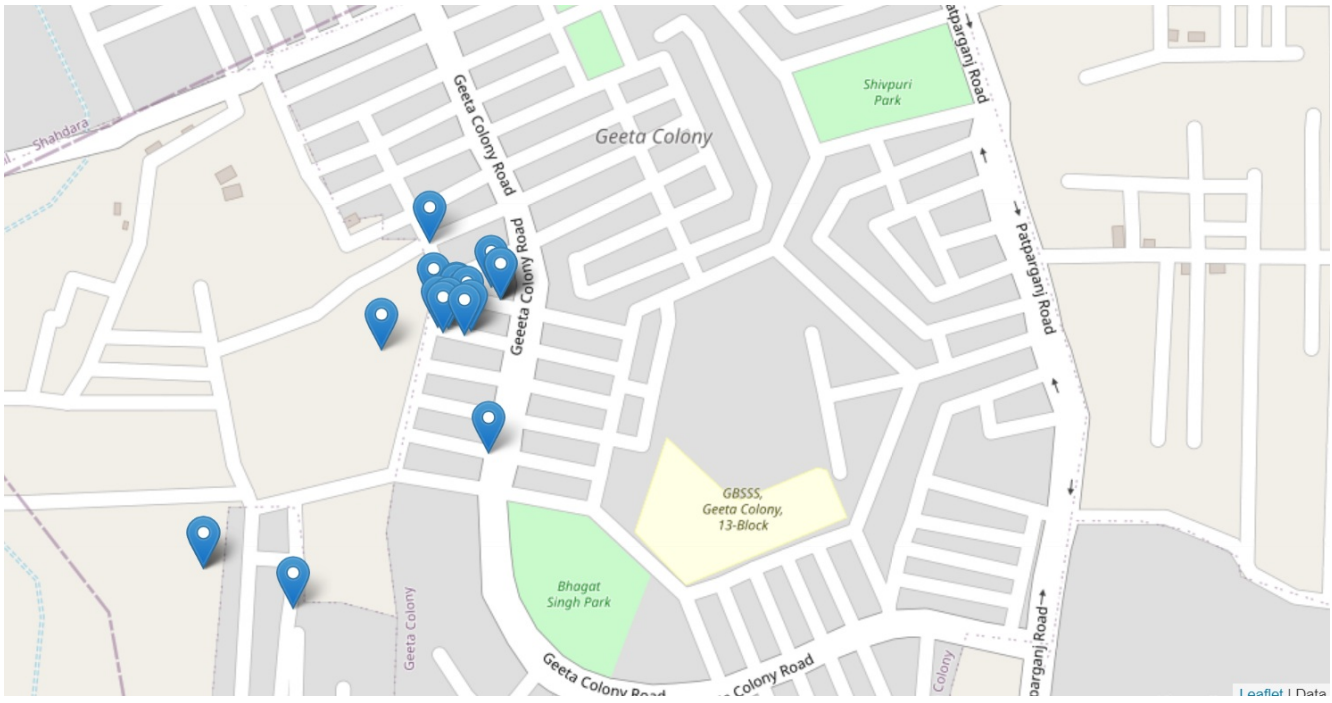
In [56]:

```
# displaying first 20 locations using folium library.
import folium
map_f = folium.Map(
    location=[28.5011226, 77.4099794],
    zoom_start=12)
l1.apply(lambda row:folium.Marker(location=[row["Latitude"], row["Longitude"]]).add_to(map_f), axis
=1)
map_f.save("map_f.html")
```

In [60]:

```
from IPython.display import Image
Image(filename='map.png')
```

Out[60]:



In [ ]:

```
```