MongoDB

# **Introducción Mongodb**

En este apartado se explicarán los pasos seguidos para cargar los datos en una base de datos Mongodb.

El proceso ha pasado por varias fases, la primera es el análisis de los diferentes tipos de fichero que ofrece la página de datos públicos de la ciudad de San Francisco, para poder elegir el que mejor se adapte. Los principales tipos son json y csv, este ultimo ha sido el elegido por el tamaño reducido que tiene en comparación con json.

En segundo lugar, se ha realizado una limpieza del dataset, haciendo los cambios necesarios en las columnas que no estén en un formato adecuado para poder explotar su contenido. Por último, se ha realizado la inserción de los datos en la base de datos.

Como una ampliación se ha descargado un dataset[[1]](#footnote-1) que contiene la división de la ciudad de San Francisco en polígonos.

En el apartado siguiente se va a detallar el proceso de **limpieza y carga** (de aquí adelante referido como **LC**) del dataset de incidencias en la base de datos Mongodb.

# **Proceso LC**

Se ha usado el lenguaje de programación *Python* para realizar este proceso por la rapidez de desarrollo y la facilidad de manejo de grandes cantidad datos que permite usando la librería *Pandas*.

El proceso de LC se ha realizado en el mismo fichero Python *process\_insert.py*, a continuación, se explican las partes mas importantes del código que se ve en la figura 1.

La función *import\_content* en la línea 31 es la que arranca el proceso LC, recibe la ruta del fichero csv que contiene el dataset. Esta función realiza llamadas a otras funciones auxiliares.

En primer lugar, se crear una conexión a servicio de Mongodb local, y en la línea 33 creamos la base de datos a la que llamamos *san\_francisco\_incidents*.

El siguiente paso es crear una colección[[2]](#footnote-2) que representara los documentos[[3]](#footnote-3) de las incidencias en la base de datos. En este caso, hemos usado una colección para representar las incidencias, además de una colección para representar los polígonos que componen los distritos de San Francisco, esta colección se explicará más adelante en último capítulo de este documento.

En la línea 36, usando la librería *panda*, leemos el fichero entero en memoria usando la función *read\_csv* que devuelve un dataframe con todo el dataset.

A continuación, procedemos con la limpieza del dataframe, empezando con las columnas X i Y representando la longitud y latitud consecutivamente. En las líneas 37 y 38 se llama a la función auxiliar *parse\_float* que recibe una variable y la convierte a float.

Luego pasamos a transformar la columna de localización (Location) que está en formato string y que normalmente no es lo más adecuado para poder realizar consultas geoespaciales sobre mongodb. Entonces en la línea 40 se llama a la función *parse\_location* pasándolo cada fila del dataframe, y convierte el contenido de la columna Location de formato string a formato *GeoJSON* en la figura 3 se ve la columna Location con el nuevo formato.

Una parte muy importante para el análisis es el tiempo. En este dataset, existen dos columas, Date y Time, que contiene por separado la fecha y la hora del incidente. Nos conviene mejor tener estos dos datos en conjunto por lo que usamos la función auxiliar *parse\_complete\_date* en la línea 42 que recibe una del dataframe y cambia en ella el campo Date, haciendo que contenga toda la información del tiempo del incidente, en la figura 3 podemos ver como queda reflejada toda esa información en el mismo campo.

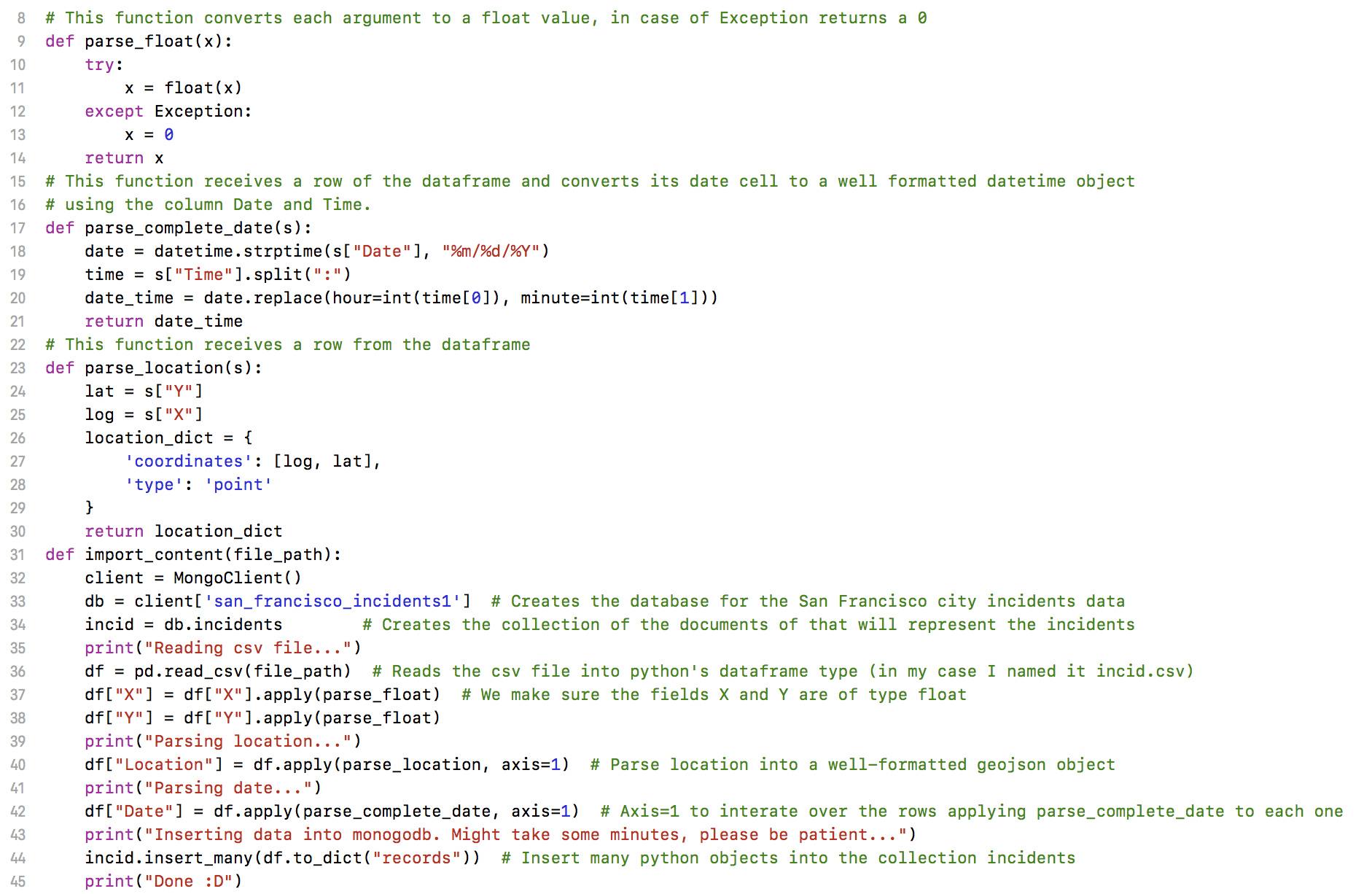


Figura Función que recibe la ruta del fichero del dataset y realiza el proceso LC

# **Resultado proceso LC**

Después de este proceso de limpieza y carga, el resultado que se ha obtenido es que hemos pasado de tener los datos representados en el formato que se ve en la figura 2 al formato en la figura 3, que tiene mucha más flexibilidad y posibilidades de explotación para extraer información.

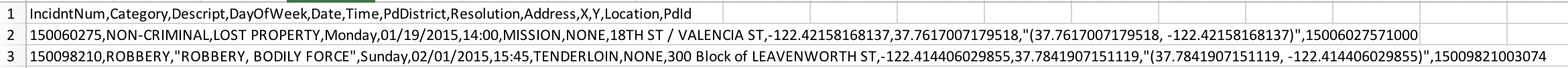


Figura dos entradas de incidencias en formato csv



Figura los mismos dos registros cambiados a documentos de Mongodb

Un punto interesante en las capturas anteriores que hemos notado, es el cambio en el tiempo en el documento en la figura 3. Como se puede ver que la hora de incidente son 14:00, pero en el campo Date nos sale que es las 15:00. La explicación es que programa usado para mostrar esos datos, Mongodb Compass, añade una hora por la zona horaria en la que estamos.

En realidad, el campo Date es correcto a nivel de base de datos y eso se comprueba con la siguiente captura que muestra una consulta para obtener un documento, y que devuelve el primero que se corresponde con el documento de la figura 3 en la izquierda.



Figura Consulta para obtener un documento (el primero por defecto)

# **Ampliación con dataset de distritos de San Francisco**

En este apartado se explicará el proceso seguido para añadir a la base de datos una nueva colección que va a representar los diferentes distritos de la ciudad de San Francisco.

El objetivo de esta ampliación es poder realizar consultas geo-espaciales que permiten tener información más detallada y exacta sobre los incidentes.

## **Análisis del dataset**

Esta vez se ha usado un fichero json en vez de un csv. Por lo que se ha desarrollado un nuevo script para realizar el proceso de transformación y carga de datos.

En primer lugar, se ha analizado detenidamente el contenido del json, y se ha logrado identificar las claves en el json que se necesitan para realizar las siguientes tareas:

1. Obtener los nombres de los campos que formaran el documento en la colección de Mongodb.
2. Obtener los datos.
3. Fusionar la lista de nombres de campos y la lista de datos en un dataframe de la librería pandas.
4. Cargar el dataframe en la base de datos después de realizar formateos si hacen falta.

La figura 5 muestra el código usado, nos basaremos sobre ella para explicar los pasos.

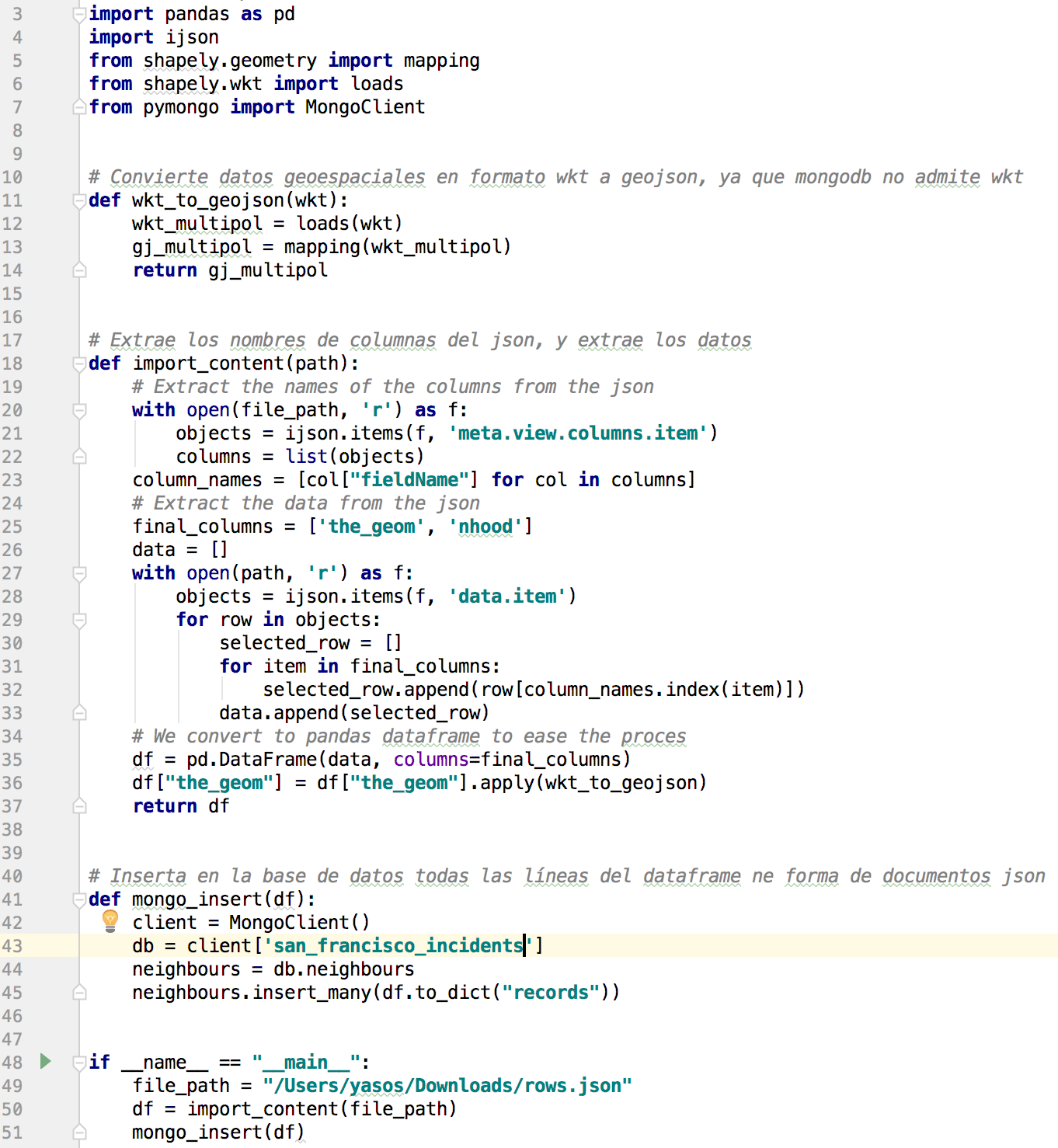


Figura 5 código para procesar y insertar el dataset de distritos de San Francisco

### **Obtención de nombres de campos.**

Viendo la estructura del json, figura 6, se ha visto que la clave “columns” dentro de la jerarquía del json, contiene un array de atributos. Entre estos atributos encontramos el campo fieldName que contiene el nombre del campo.

Entre las líneas 20 y 25, leemos el fichero, extraemos de cada elemento del array de columns, el nombre del campo.

En nuestro caso, nos interesan sólo dos campos, él de los datos de geo-localización y él del nombre del distrito, por lo que definimos una lista (línea 25), con los nombres de esos campos, esa lista nos servirá más adelante para extraer solo los datos que nos interesan.



Figura 6 json del dataset de distritos

### **Obtención de datos en un dataframe, transformaciones y carga en base de datos.**

Entre las líneas 27 y 35, se procesa el fichero json para recuperar los valores de los dos campos que nos interesan del dataset (the\_geom, nhood).

En la línea 36 realizamos una transformación sobre el campo the\_geom que contiene los polígonos que forman los distritos de la ciudad, esa transformación consiste en convertir los datos del formato WKT a GeoJson usando la liberaría *shapely* de Python. La transformación se realiza sobre cada fila del dataframe usando la función auxiliar wkt\_to\_geojson.

Comparando las figuras 7 y 8 se puede ver la diferencia en el formato de ese campo.

Al realizar las transformaciones, la inserción en Mongodb es inmediata, convirtiendo el dataframe a un diccionario de Python, como se ve en la línea 45.

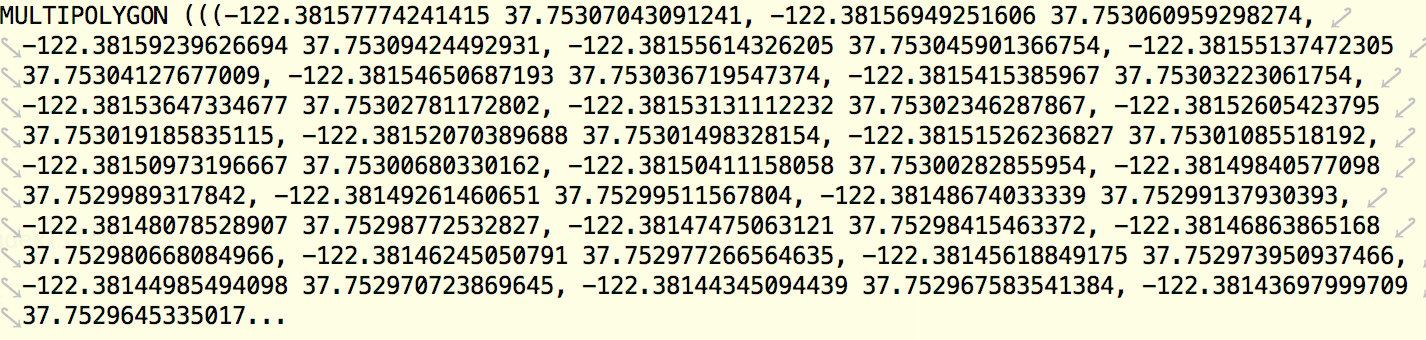


Figura the\_geom en formato wkt

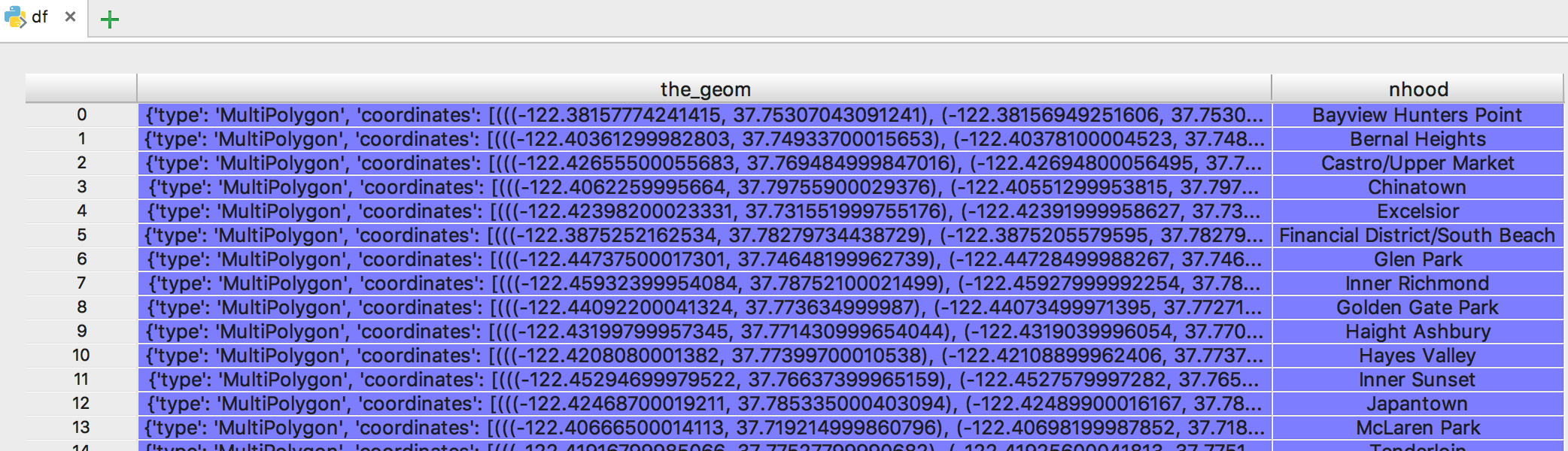


Figura dataframe resultante al final de las transformaciones, the\_geom en formato geojson

1. El enlace de descarga https://data.sfgov.org/Geographic-Locations-and-Boundaries/SF-Find-Neighborhoods/pty2-tcw4 [↑](#footnote-ref-1)
2. Una colección es equivalente a una tabla de base de datos en los sistemas relacionales. [↑](#footnote-ref-2)
3. Los datos en mongodb se guardan en documentos, un documento equivale a un registro en las bases de datos relacionales. [↑](#footnote-ref-3)