OSM Data Analysis - Ujjwal Baral

Overview:

This document reflects my personal journey in data analysis, where I've combined my insights, skills, and hands-on experience to derive meaningful conclusions The Key Steps are:

1. Gather Data Using Osmium Command Line
2. Process Data with Pandas
3. Create Basic Visualizations

1. Gather Data Using Osmium Command Line

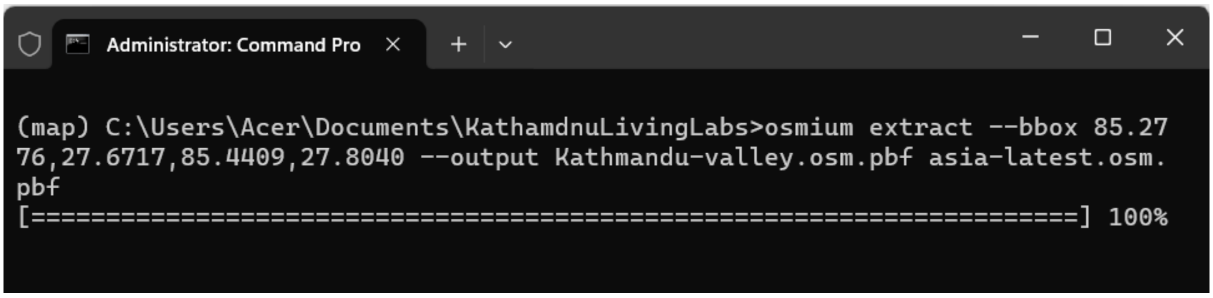
Download OSM data for Nepal from Geofabrik.

- Download asia-latest.osm.pbf-12GB.

Extract Kathmandu Valley from asia-latest.osm.pbf

```
# Bounding box format: min_lon, min_lat, max_lon, max_lat
osmium extract ^
  --bbox 85.2776,27.6717,85.4409,27.8040 ^
  --output Kathmandu-valley.osm.pbf ^
  asia-latest.osm.pbf
```

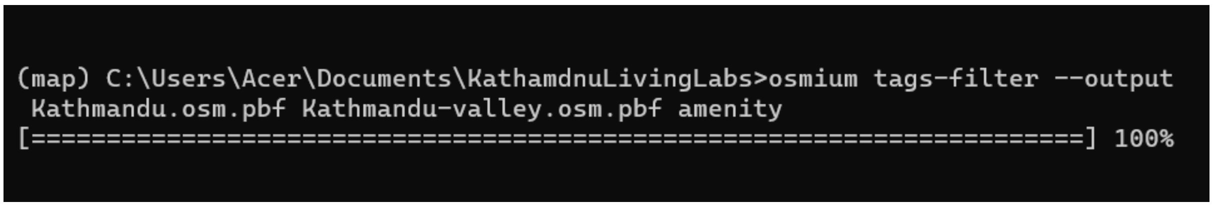
💡 Kathmandu bounding box format is 85.2776,27.6717,85.4409,27.8040



```
(map) C:\Users\Acer\Documents\KathamdnuLivingLabs>osmium extract --bbox 85.27
76,27.6717,85.4409,27.8040 --output Kathmandu-valley.osm.pbf asia-latest.osm.
pbf
[======================================================================] 100%
```

Filter OSM data by tag amenity which is not null:

```
osmium tags-filter --output Kathmandu.osm.pbf Kathmandu-valley.osm.pbf amenit
```



```
(map) C:\Users\Acer\Documents\KathamdnuLivingLabs>osmium tags-filter --output
 Kathmandu.osm.pbf Kathmandu-valley.osm.pbf amenity
[======================================================================] 100%
```

Now we have Kathmandu.osm.pbf file.

2. Process Data with Pandas

To read OpenStreetMap (OSM) data in PBF format into a Pandas Data Frame, you can use the `osmium` library,

Install Package

```
#for osm
!pip install osmium
#for language-translation
!pip install -U deep-translator
#for map visualization
!pip install folium
!pip install seaborn
!pip install matplotlib
#for coordinates distance calculation
!pip install geopy
```

Create a Handler

```
import osmium as osm
import pandas as pd

class OSMHandler(osm.SimpleHandler):
    def __init__(self):
        osm.SimpleHandler.__init__(self)
        self.osm_data = []

    def tag_inventory(self, elem, elem_type):
        coordinates = None

        if elem_type == "node":
            coordinates = f"{elem.location.lat}/{elem.location.lon}"

        for tag in elem.tags:
            self.osm_data.append([elem_type,
                                   elem.id,
                                   elem.version,
                                   elem.visible,
                                   pd.Timestamp(elem.timestamp),
                                   elem.uid,
                                   elem.user,
                                   elem.changeset,
                                   len(elem.tags),
                                   coordinates,
                                   tag.k,
                                   tag.v])

    def node(self, n):
        self.tag_inventory(n, "node")

    def way(self, w):
        self.tag_inventory(w, "way")

    def relation(self, r):
        self.tag_inventory(r, "relation")

osmhandler = OSMHandler()
# scan the input file and fill the handler list accordingly
osmhandler.apply_file("Kathmandu.osm.pbf")

# transform the list into a pandas DataFrame
data_colnames = ['type', 'id', 'version', 'visible', 'ts', 'uid',
                 'user', 'chgset', 'ntags', 'coordinates', 'tagkey', 'tagvalue']

# create Dataframe(df)
df_osm = pd.DataFrame(osmhandler.osm_data, columns=data_colnames)

# Corrected line to sort the DataFrame
df_osm = df_osm.sort_values(by=['type', 'id', 'ts'])
#show top 10 dataframe
df_osm.head(10)
```

	type	id	version	visible	ts	uid	user	chgset	ntags	coordinates	tagkey	tagvalue	is_duplicate
0	node	268310351	10	True	2023-05-25 10:02:57+00:00	0		0	5	27.7353517/85.3057238	amenity	bus_station	False
1	node	268310351	10	True	2023-05-25 10:02:57+00:00	0		0	5	27.7353517/85.3057238	name	Macha Pokhari Bus Station	False
2	node	268310351	10	True	2023-05-25 10:02:57+00:00	0		0	5	27.7353517/85.3057238	name:en	Pasang Lhamu Ticket Counter - Trishuli, Dhunch...	False
3	node	268310351	10	True	2023-05-25 10:02:57+00:00	0		0	5	27.7353517/85.3057238	opening_hours	Mo-Su 05:00-18:00	False
4	node	268310351	10	True	2023-05-25 10:02:57+00:00	0		0	5	27.7353517/85.3057238	phone	014356342	False
5	node	279376624	4	True	2017-03-09 04:41:46+00:00	0		0	4	27.7067985/85.314477	amenity	bus_station	False
6	node	279376624	4	True	2017-03-09 04:41:46+00:00	0		0	4	27.7067985/85.314477	created_by	Potlatch 0.10f	False
7	node	279376624	4	True	2017-03-09 04:41:46+00:00	0		0	4	27.7067985/85.314477	name:en	Ratna Park bus station	False
8	node	279376624	4	True	2017-03-09 04:41:46+00:00	0		0	4	27.7067985/85.314477	name:zh	去巴德岗，去帕坦	False
9	node	279376643	1	True	2008-07-18 22:00:52+00:00	0		0	2	27.7158423/85.2892175	amenity	place_of_worship	False

Check Rows and Columns

```
df_osm.shape
```

```
(59745, 12)
```

Check Duplicates rows.

- Add `is_duplicate` column as a duplicate flag.

```
df_osm['is_duplicate'] = df_osm.duplicated(['type', 'id','tagkey','tagvalue'], keep=False)
df_osm.head(10)
```

💡 On the basis of `type` , `id` column there is multiple duplicate which later will be handled by pivoting the data frame. For now, there is no duplicates as for required column.

Show `tagkey` column value with each count

```
tagkey_counts = df_osm['tagkey'].value_counts()
# Print the top 20 values
print(tagkey_counts.head(20))
```

```
amenity             13472
name                10101
name:en              3174
source               2365
name:ne              1602
operator:type        1559
addr:street          1534
building_count       1532
personnel:count      1508
phone                1401
isced:level          1327
student:count        1316
operator             1081
building             1063
opening_hours         968
religion              928
addr:city             682
wheelchair            592
cuisine               483
toilets:wheelchair    427
Name: tagkey, dtype: int64
```

Copy the highest value.

- Choose the value who have highest count.

- Select Some name value as for analysis.

```
# df_osm is the original DataFrame
df = df_osm[df_osm['tagkey'].isin(['amenity', 'name','name:ne','name:en', 'source','capacity'])].copy()

# Display the  DataFrame
df.sample(10)
```

	type	id	version	visible	ts	uid	user	chgset	ntags	coordinates	tagkey	tagvalue	is_duplicate
24564	node	9942141117	1	True	2022-08-09 09:49:47+00:00	0		0	2	27.6730332/85.3243908	amenity	cafe	False
50747	way	225594252	7	True	2013-10-18 08:50:39+00:00	0		0	31	None	amenity	school	False
54330	way	245075713	2	True	2014-05-19 11:58:40+00:00	0		0	3	None	name	Be There	False
19081	node	7088157087	3	True	2023-03-22 10:59:19+00:00	0		0	4	27.7161811/85.3459386	name:en	Shree Pashupati Secondary School	False
4428	node	2168874678	7	True	2023-06-15 09:50:37+00:00	0		0	5	27.7270135/85.3548968	name:ne	सरस्वती मन्दिर	False
58929	way	1125359900	2	True	2022-12-29 09:04:23+00:00	0		0	4	None	amenity	motorcycle_parking	False
48133	way	223275407	4	True	2013-08-26 10:21:35+00:00	0		0	8	None	source	OpenDRI survey	False
54789	way	313548989	2	True	2021-02-15 06:42:56+00:00	0		0	5	None	amenity	parking	False
8678	node	4264083856	4	True	2023-06-08 11:23:35+00:00	0		0	5	27.6984576/85.3311776	name:en	Mandir	False
22863	node	9529746968	2	True	2023-07-08 14:33:06+00:00	0		0	3	27.7021793/85.310755	amenity	bank	False

Pivot the Data frame.

- Convert `tagkey` row to Column with `tagvalue` assign with it.
- Select Some name value as for analysis.

```
pivoted_df = df.pivot_table(index=['type', 'id', 'version', 'visible', 'ts', 'uid', 'coordinates', 'chgset', 'ntags'],
                                            columns='tagkey', values='tagvalue', aggfunc='first').reset_index()

# Display the pivoted DataFrame
pivoted_df.sample(10)
```

tagkey	type	id	version	visible	ts	uid	coordinates	chgset	ntags	amenity	capacity	name	name:en	name:ne	source
3569	node	4753802823	2	True	2017-12-30 17:27:13+00:00	0	27.7348577/85.3096909	0	2	bus_station	NaN	NaN	Bus to Pokhara #bor	NaN	NaN
9438	node	10048819945	1	True	2022-09-24 10:24:51+00:00	0	27.6850951/85.3664544	0	2	pharmacy	NaN	Zenith Pharmacy	NaN	NaN	NaN
6860	node	9599541321	1	True	2022-03-23 11:34:31+00:00	0	27.7023503/85.3597167	0	5	toilets	NaN	NaN	NaN	NaN	NaN
6073	node	8806897302	3	True	2023-06-15 09:50:37+00:00	0	27.7059925/85.2974063	0	4	water_point	NaN	चागलधारा	Chagal Dhara	चागलधारा	NaN
6617	node	9528551017	2	True	2022-09-11 19:12:33+00:00	0	27.7413604/85.3299974	0	2	restaurant	NaN	CFC THE AIRPORT SEKUWA CORNER	NaN	NaN	NaN
137	node	1498876026	4	True	2020-03-30 10:51:33+00:00	0	27.6961409/85.3080938	0	2	cafe	NaN	Bakery Cafe Teku	NaN	NaN	NaN
3630	node	4763981721	3	True	2023-03-22 10:47:02+00:00	0	27.7053643/85.3122083	0	6	place_of_worship	NaN	गणेश मन्दिर	Ganesh Temple	गणेश मन्दिर	NaN
602	node	1937721989	1	True	2012-09-29 09:28:13+00:00	0	27.6728786/85.3179075	0	2	ngo	NaN	Non-governmental Organization Nepal	NaN	NaN	NaN
1028	node	2074412129	3	True	2015-07-17 06:44:08+00:00	0	27.7260665/85.3233732	0	2	atm	NaN	NaN	NaN	NaN	NaN
8273	node	10011219686	1	True	2022-09-09 12:03:18+00:00	0	27.7354667/85.3079795	0	2	bar	NaN	Fulchoki Dance Bar	NaN	NaN	NaN

Merge the Column

- Merge `name` , `name:en,` `name:ne` to single column `name`
- And drop that existing column.

```
pivoted_df['name'] = pivoted_df['name'].fillna(pivoted_df['name:en']).fillna(pivoted_df['name:ne'])

# Drop 'name:en' and 'name:ne' columns
```

```
pivoted_df.drop(['name:en', 'name:ne'], axis=1, inplace=True)
pivoted_df.sample(10)
```

tagkey	type	id	version	visible	ts	uid	coordinates	chgset	ntags	amenity	capacity	name	source
8979	node	10016425399	1	True	2022-09-11 18:22:10+00:00	0	27.7494371/85.3361572	0	2	restaurant	NaN	Gaule Khaja Ghar	NaN
10376	node	10611337612	2	True	2023-06-29 05:56:08+00:00	0	27.7077741/85.3833034	0	2	community_centre	NaN	Jestha Nagharik Building	NaN
8485	node	10013011010	1	True	2022-09-10 10:37:02+00:00	0	27.7380743/85.3096627	0	2	restaurant	NaN	Dhangadhi kanchanpur khaja Ghar	NaN
3855	node	4834152622	1	True	2017-05-04 08:01:05+00:00	0	27.7393761/85.3365784	0	3	bus_station	NaN	Chakrapath	NaN
2911	node	4329327889	1	True	2016-07-31 11:32:24+00:00	0	27.6728835/85.3245667	0	2	cafe	NaN	Dani's Handmade Coffee	NaN
685	node	1962367558	2	True	2023-03-22 11:13:56+00:00	0	27.7178482/85.3363133	0	5	place_of_worship	NaN	हाडिगाउँ भुटेश्वर मन्दिर	NaN
3311	node	4577903489	1	True	2016-12-28 17:54:09+00:00	0	27.7250073/85.31523	0	3	fast_food	NaN	NaN	NaN
4900	node	5995392686	1	True	2018-10-18 12:15:20+00:00	0	27.7206299/85.361395	0	1	bureau_de_change	NaN	NaN	NaN
3572	node	4754370527	1	True	2017-03-25 11:44:29+00:00	0	27.678172/85.3152943	0	2	restaurant	NaN	Minas Bhojanalaya	NaN
8188	node	10010484025	1	True	2022-09-09 05:16:29+00:00	0	27.7367479/85.3224118	0	2	restaurant	NaN	Lumbini Tandoori Dawa And Bhojanalaya	NaN

Check the Datatypes

After selecting the columns, we start addressing the quality and consistency issues in the important columns.

```
pivoted_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10702 entries, 0 to 10701
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   type         10702 non-null  object
 1   id           10702 non-null  int64
 2   version      10702 non-null  int64
 3   visible      10702 non-null  bool
 4   ts           10702 non-null  datetime64[ns, UTC]
 5   uid          10702 non-null  int64
 6   coordinates  10702 non-null  object
 7   chgset       10702 non-null  int64
 8   ntags        10702 non-null  int64
 9   amenity      10193 non-null  object
 10  capacity     37 non-null     object
 11  name         8743 non-null   object
 12  source       768 non-null    object
dtypes: bool(1), datetime64[ns, UTC](1), int64(5), object(6)
memory usage: 1013.9+ KB
```

Handle `capacity` column.

The `capacity` column is expected to contain integer values, But First:

- We need to inspect its unique values and address any inconsistencies in the data.

```
capacities = pivoted_df['capacity'].unique()
print(capacities)
```

```
[nan '0' '100' '20' '30' '150' '50' '35-40' '200' '25' '60' '100 plus'
 '15' '35' '10' '40' '86' '7' '600' '15 cars and 40 bikes' '40-50']
```

- Handle string, null value and convert it to `int` type.

```
pivoted_df['capacity'] = pivoted_df['capacity'].replace('100 plus', 100)
pivoted_df['capacity'] = pivoted_df['capacity'].replace('15 cars and 40 bikes',40)
pivoted_df['capacity'] = pivoted_df['capacity'].replace('40-50',50)
```

```
#  convert it to numeric
pivoted_df['capacity'] = pd.to_numeric(pivoted_df['capacity'], errors='coerce')

# Specify a default value for NaN (replace NaN with 0)
pivoted_df['capacity'] = pivoted_df['capacity'].fillna(0).astype(int)


capacities = pivoted_df['capacity'].unique()

#check unique
print(capacities)

#check the capacity column data type
print(pivoted_df['capacity'].dtype)
```

```
[  0 100  20  30 150  50 200  25  60  15  35  10  40  86   7 600]
int64
```

Handle `amenity` column.

Check any inconsistent value in `amenity` column.

- We inspect its unique values and address any inconsistencies in the data.

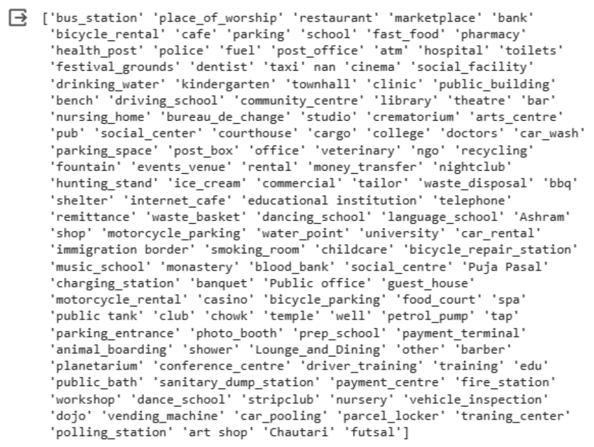
```
import re

# Convert to lowercase and get unique values
pivoted_df['amenity'] = pivoted_df['amenity'].str.lower()
unique_amenities = pivoted_df['amenity'].unique()

# Define a pattern to check for inconsistent characters
pattern = re.compile(r'[;:/!@]')

# Check for inconsistent characters in unique_amenities
inconsistent_amenities = [amenity for amenity in unique_amenities if pattern.search(amenity)]

# Display the inconsistent amenities
print("Inconsistent Amenities:")
print(inconsistent_amenities)
```

```
Inconsistent Amenities:
['doctors;clinic', 'office;restaurant', 'toilets;bank']
```

There is `office;restaurant` , `doctors;clinic` , `toilets;bank` values in `amenity` which are inconsistent.

- Address the inconsistent values by splitting them.

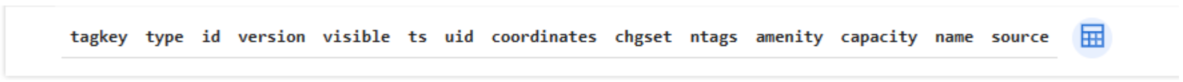
```
pivoted_df['amenity'] = pivoted_df['amenity'].apply(lambda x: x.split(';')[0] if pd.notna(x) else x)
unique_amenities = pivoted_df['amenity'].unique()
print(unique_amenities)
```



```
['bus_station' 'place_of_worship' 'restaurant' 'marketplace' 'bank'
 'bicycle_rental' 'cafe' 'parking' 'school' 'fast_food' 'pharmacy'
 'health_post' 'police' 'fuel' 'post_office' 'atm' 'hospital' 'toilets'
 'festival_grounds' 'dentist' 'taxi' nan 'cinema' 'social_facility'
 'drinking_water' 'kindergarten' 'townhall' 'clinic' 'public_building'
 'bench' 'driving_school' 'community_centre' 'library' 'theatre' 'bar'
 'nursing_home' 'bureau_de_change' 'studio' 'crematorium' 'arts_centre'
 'pub' 'social_center' 'courthouse' 'cargo' 'college' 'doctors' 'car_wash'
 'parking_space' 'post_box' 'office' 'veterinary' 'ngo' 'recycling'
 'fountain' 'events_venue' 'rental' 'money_transfer' 'nightclub'
 'hunting_stand' 'ice_cream' 'commercial' 'tailor' 'waste_disposal' 'bbq'
 'shelter' 'internet_cafe' 'educational institution' 'telephone'
 'remittance' 'waste_basket' 'dancing_school' 'language_school' 'Ashram'
 'shop' 'motorcycle_parking' 'water_point' 'university' 'car_rental'
 'immigration border' 'smoking_room' 'childcare' 'bicycle_repair_station'
 'music_school' 'monastery' 'blood_bank' 'social_centre' 'Puja Pasal'
 'charging_station' 'banquet' 'Public office' 'guest_house'
 'motorcycle_rental' 'casino' 'bicycle_parking' 'food_court' 'spa'
 'public tank' 'club' 'chowk' 'temple' 'well' 'petrol_pump' 'tap'
 'parking_entrance' 'photo_booth' 'prep_school' 'payment_terminal'
 'animal_boarding' 'shower' 'Lounge_and_Dining' 'other' 'barber'
 'planetarium' 'conference_centre' 'driver_training' 'training' 'edu'
 'public_bath' 'sanitary_dump_station' 'payment_centre' 'fire_station'
 'workshop' 'dance_school' 'stripclub' 'nursery' 'vehicle_inspection'
 'dojo' 'vending_machine' 'car_pooling' 'parcel_locker' 'traning_center'
 'polling_station' 'art shop' 'Chautari' 'futsal']
```

Check any duplicates rows incase:

```
pivoted_df[pivoted_df.duplicated(subset=['id','type'], keep=False)]
```

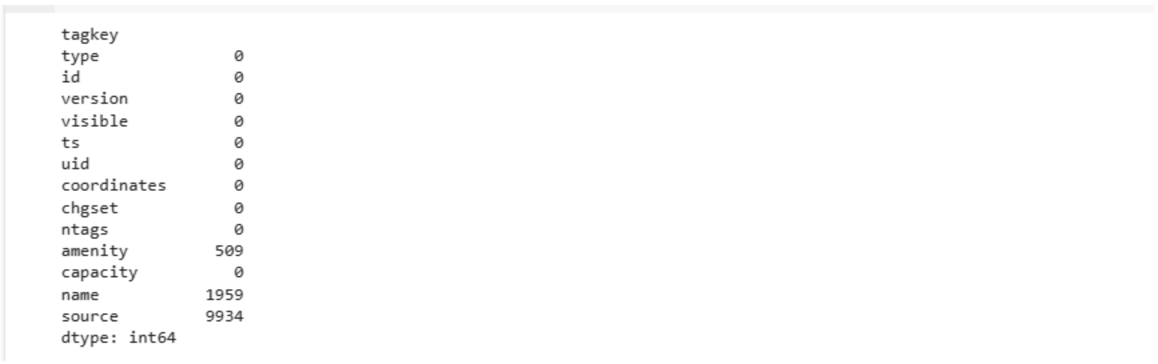


tagkey	type	id	version	visible	ts	uid	coordinates	chgset	ntags	amenity	capacity	name	source

No Duplicates Found.

Check Null Value.

```
pivoted_df.isnull().sum()
```



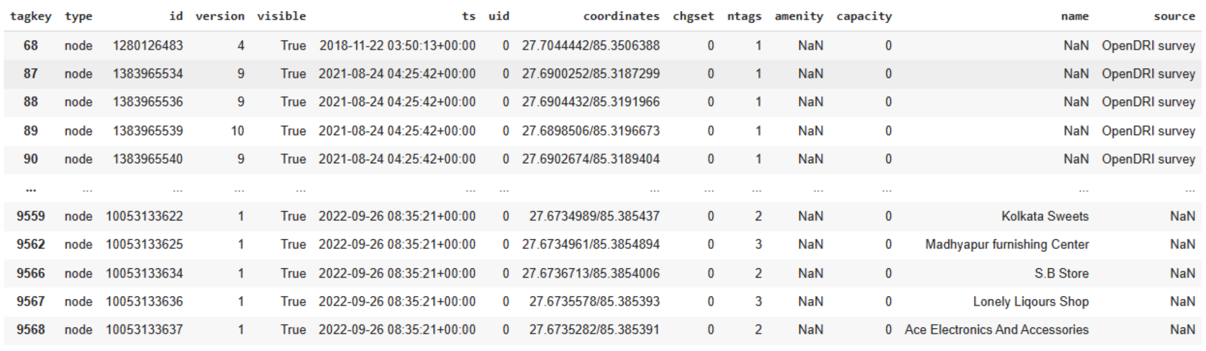
```
tagkey
type              0
id                0
version           0
visible           0
ts                0
uid               0
coordinates       0
chgset            0
ntags             0
amenity         509
capacity          0
name           1959
source         9934
dtype: int64
```

There are 3 columns (`amenity`, `name`, `source`) which consists null values. We handle this one by one.

Check `amenity` Value.

- Show null value.

```
pivoted_df[pivoted_df['amenity'].isnull()]
```



tagkey	type	id	version	visible	ts	uid	coordinates	chgset	ntags	amenity	capacity	name	source
68	node	1280126483	4	True	2018-11-22 03:50:13+00:00	0	27.7044442/85.3506388	0	1	NaN	0	NaN	OpenDRI survey
87	node	1383965534	9	True	2021-08-24 04:25:42+00:00	0	27.6900252/85.3187299	0	1	NaN	0	NaN	OpenDRI survey
88	node	1383965536	9	True	2021-08-24 04:25:42+00:00	0	27.6904432/85.3191966	0	1	NaN	0	NaN	OpenDRI survey
89	node	1383965539	10	True	2021-08-24 04:25:42+00:00	0	27.6898506/85.3196673	0	1	NaN	0	NaN	OpenDRI survey
90	node	1383965540	9	True	2021-08-24 04:25:42+00:00	0	27.6902674/85.3189404	0	1	NaN	0	NaN	OpenDRI survey
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9559	node	10053133622	1	True	2022-09-26 08:35:21+00:00	0	27.6734989/85.385437	0	2	NaN	0	Kolkata Sweets	NaN
9562	node	10053133625	1	True	2022-09-26 08:35:21+00:00	0	27.6734961/85.3854894	0	3	NaN	0	Madhyapur furnishing Center	NaN
9566	node	10053133634	1	True	2022-09-26 08:35:21+00:00	0	27.6736713/85.3854006	0	2	NaN	0	S.B Store	NaN
9567	node	10053133636	1	True	2022-09-26 08:35:21+00:00	0	27.6735578/85.385393	0	3	NaN	0	Lonely Liqours Shop	NaN
9568	node	10053133637	1	True	2022-09-26 08:35:21+00:00	0	27.6735282/85.385391	0	2	NaN	0	Ace Electronics And Accessories	NaN

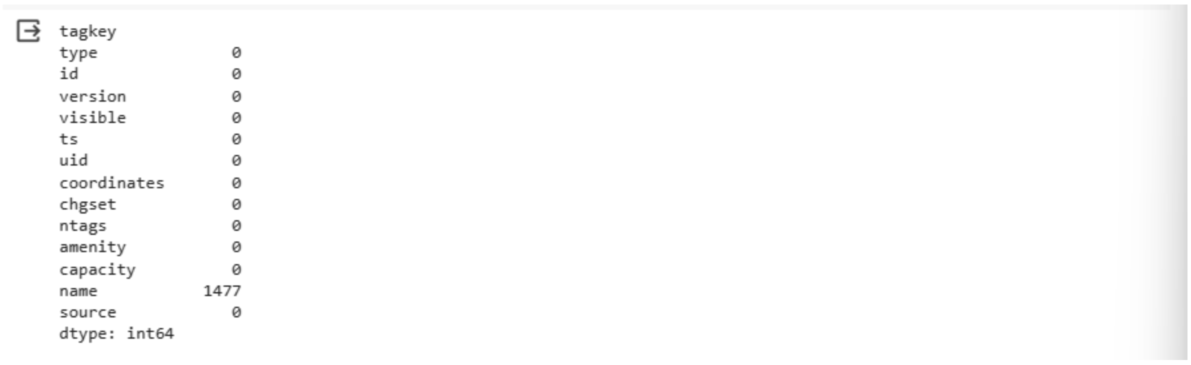
- Drop the null value.

```
pivoted_df.dropna(subset=['amenity'], inplace=True)

#check the null value
pivoted_df['amenity'].isnull().sum()

>> 0
```

Check `source` value.

- Handle Null by filling `-` Value.

```
pivoted_df['source'].fillna('-', inplace=True)

#check the total null value
pivoted_df.isnull().sum()
```



```
tagkey
type              0
id                0
version           0
visible           0
ts                0
uid               0
coordinates       0
chgset            0
ntags             0
amenity           0
capacity          0
name           1477
source            0
dtype: int64
```

Rename and Rearrange the Column names.

```
# create order list
new_column_order = ['element_type', 'element_id', 'element_version', 'is_visible',
                    'timestamp', 'user_id', 'coordinates', 'num_tags', 'amenity_type',
                    'amenity_name', 'changeset', 'capacity_value', 'data_source']

# Create a dictionary to map old column names to new names
column_mapping = {
    'type': 'element_type',
    'id': 'element_id',
    'version': 'element_version',
    'visible': 'is_visible',
    'ts': 'timestamp',
    'uid': 'user_id',
    'coordinates': 'coordinates',
    'chgset': 'changeset',
    'ntags': 'num_tags',
    'amenity': 'amenity_type',
    'capacity': 'capacity_value',
    'name': 'amenity_name',
    'source': 'data_source'
}

# Rename columns
pivoted_df = pivoted_df.rename(columns=column_mapping)

# Create new df and reorder columns and drop null
df = pivoted_df[new_column_order]
df.dropna()

#print the column
print(df.columns)
```

```
Index(['element_type', 'element_id', 'element_version', 'is_visible',
       'timestamp', 'user_id', 'coordinates', 'num_tags', 'amenity_type',
       'amenity_name', 'changeset', 'capacity_value', 'data_source'],
      dtype='object', name='tagkey')
```

Translate `name` from Nepali to English

- Extract Nepali Language from `name` column to list.

```
import re

# Pattern to match non-English characters (Nepali/Hindi)
Nepali_pattern = re.compile(r'[^\u0000-\u007F]+')

# Filter rows where 'amenity_name' contains non-English characters
Nepali_rows = df[df['amenity_name'].str.contains(Nepali_pattern, na=False)]

# Store the 'amenity_name' values in a list
Nepali_list = Nepali_rows['amenity_name'].tolist()

# Count the list
print(len(Nepali_list))
```

```
1583
```

- Select Unique from the list

```
unique_nepali_list = list(set(Nepali_list))

# Print the unique Hindi values
print(len(unique_nepali_list))
print(unique_nepali_list)
```

```
1410
['जलख्या', 'एभरेस्ट तन्दुरी धावा', 'श्री बाल बयाबासाये केन्द्र माध्यमिक विद्यालय', 'स्थानीय रेस्ट्रो र बार', 'Dé Temple Cafe', 'नेपाल इन्भेष्टमेन्ट बैंकको एटीएम', 'लखै हाउस', 'Sanima Bank सानीमा बैंक',
```

- Use Translator

```
from deep_translator import GoogleTranslator


# Use GoogleTranslator to translate each text in the list
translations_to_english = [GoogleTranslator(source='auto', target='en').translate(text) for text in unique_nepali_list]
```

```
# Print the translations
print(translations_to_english)
```

💡 Note: The execution time for the process ranges from 3 to 10 minutes.

```
['Jalakhya', 'Everest Tandoori Rush', 'Shri Bal Byabasaye Center Secondary School', 'Local restaurants and bars', 'Dé Temple Cafe', 'Nepal Investment Bank ATM',
```

- Replace the Nepali list with its corresponding English list in the `name` column dataframe.

```
df['amenity_name']=df['amenity_name'].replace(unique_nepali_list,translations_to_english)

#display the data
df.sample(10)
```

tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value	data_source
7807	node	9974264435	1	True	2022-08-24 07:16:46+00:00	0	27.6735258/85.3564264	2	events_venue	Indrasan Banquet and Restaurant	0	0	-
5376	node	6501663544	2	True	2023-04-26 09:22:47+00:00	0	27.7162741/85.427897	3	place_of_worship	Shiv Temple	0	0	-
6686	node	9528662619	1	True	2022-02-23 02:12:21+00:00	0	27.7173785/85.346268	2	dentist	Omkar Dental home	0	0	-
7362	node	9942220322	1	True	2022-08-09 10:47:30+00:00	0	27.6739843/85.3249383	5	restaurant	NewaribKitchen	0	0	-
10485	node	10803756805	1	True	2023-04-12 06:18:12+00:00	0	27.7151342/85.3127303	2	fast_food	Western Tandoori	0	0	-
3946	node	4908201926	1	True	2017-06-10 22:48:38+00:00	0	27.7007034/85.3082523	2	drinking_water	S.S. Water Supply	0	0	-
8724	node	10015291520	1	True	2022-09-11 08:00:55+00:00	0	27.740472/85.3262838	2	restaurant	SUJAN FAST FOOD	0	0	-
6098	node	8829801142	1	True	2021-06-13 16:17:36+00:00	0	27.6734527/85.4186831	2	water_point	-	0	0	-
6747	node	9529746997	1	True	2022-02-23 12:06:54+00:00	0	27.7029036/85.310924	2	bank	Mega Bank Nepal Limited	0	0	-
4715	node	5686874522	1	True	2018-06-13 06:56:38+00:00	0	27.7688298/85.2983395	2	school	Guiness public	0	0	-

Handle `amenity_name` column

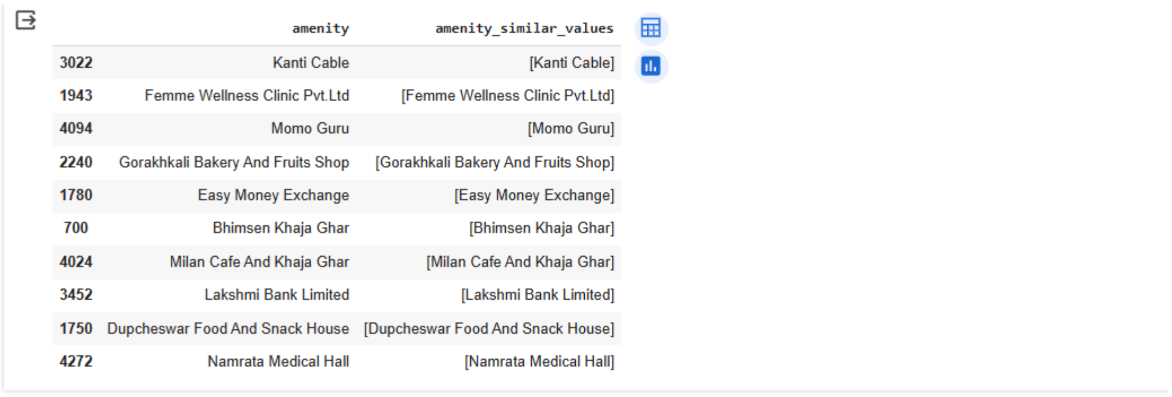
- **Format text data.**

```
# Convert the 'amenity_name' column to title case
df['amenity_name'] = df['amenity_name'].str.title()

# Remove symbols, extra whitespace, and periods
df['amenity_name'] = df['amenity_name'].str.replace(r'[^a-zA-Z0-9\s.]', '', regex=True)
df['amenity_name'] = df['amenity_name'].str.strip()

df.sample(10)
```

tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value	data_source	is_duplicate
5040	node	6052416479	2	True	2018-11-23 10:06:16+00:00	0	27.709874,85.3212008	4	bank	Sanima Life Insurance	0	0	-	False
4563	node	5450219328	1	True	2018-03-03 07:47:24+00:00	0	27.6748798,85.327138	3	place_of_worship	Nan	0	0	-	True
4332	node	5172863025	5	True	2023-06-09 08:11:30+00:00	0	27.7339038,85.3781742	5	place_of_worship	Shri Ram Janaki Temple	0	0	-	False
8715	node	10015272304	1	True	2022-09-11 08:00:55+00:00	0	27.7407204,85.3270208	4	pharmacy	Keulini Pharmacy	0	0	-	False
4912	node	6003145185	3	True	2023-06-15 09:50:37+00:00	0	27.7149414,85.2838141	9	restaurant	Swayambhu Plaza Restaurant	0	0	-	False
8237	node	10010983334	3	True	2023-07-19 03:06:55+00:00	0	27.7370907,85.3221847	5	restaurant	Tanahu Khaja And Sekuwa Corner	0	0	-	False
3035	node	4397318292	2	True	2016-09-12 11:47:31+00:00	0	27.6740398,85.3187246	3	doctors	Dr Parag Karki	0	0	-	False
9842	node	10116905177	1	True	2022-10-20 16:16:27+00:00	0	27.6794095,85.4015645	2	fast_food	S S Momo Center	0	0	-	False
5759	node	8023127285	3	True	2023-03-15 11:02:46+00:00	0	27.7398634,85.3371465	4	hospital	Metro Hospital	0	0	-	False
8287	node	10011338747	1	True	2022-09-09 12:57:57+00:00	0	27.7247473,85.3574365	1	bench	Nan	0	0	-	True

- **Find Similar Values of `amenity_name`**
  - Create new df which hold `amenity_name` and similar values.

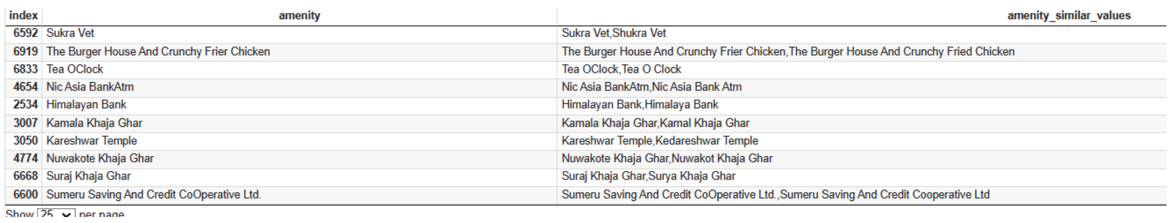
```
#using difflib library
import difflib


# Convert the 'amenity_name' column to strings
df['amenity_name'] = df['amenity_name'].astype(str)

# Get unique values and convert to DataFrame
new_df = pd.DataFrame(sorted(df['amenity_name'].unique()), columns=['amenity'])

# Function to find similar values
def find_similar(value, choices, n=2, cutoff=0.92):
    return difflib.get_close_matches(value, choices, n=n, cutoff=cutoff)

# Find similar values for each row in the 'amenity' column
new_df['amenity_similar_values'] = new_df['amenity'].apply(lambda x: find_similar(x, new_df['amenity']))

# Display the DataFrame with similar values
new_df.sample(10)
```



	amenity	amenity_similar_values
3022	Kanti Cable	[Kanti Cable]
1943	Femme Wellness Clinic Pvt.Ltd	[Femme Wellness Clinic Pvt.Ltd]
4094	Momo Guru	[Momo Guru]
2240	Gorakhkali Bakery And Fruits Shop	[Gorakhkali Bakery And Fruits Shop]
1780	Easy Money Exchange	[Easy Money Exchange]
700	Bhimsen Khaja Ghar	[Bhimsen Khaja Ghar]
4024	Milan Cafe And Khaja Ghar	[Milan Cafe And Khaja Ghar]
3452	Lakshmi Bank Limited	[Lakshmi Bank Limited]
1750	Dupcheswar Food And Snack House	[Dupcheswar Food And Snack House]
4272	Namrata Medical Hall	[Namrata Medical Hall]

- Filter which has more than 1 similar values

```
# Filter rows where 'amenity_similar_values' contains more than one list
filtered_df = new_df[new_df['amenity_similar_values'].apply(
lambda x: isinstance(x, list) and len(x) > 1)]
filtered_df.sample(20)
```



index	amenity	amenity_similar_values
6592	Sukra Vet	Sukra Vet,Shukra Vet
6919	The Burger House And Crunchy Frier Chicken	The Burger House And Crunchy Frier Chicken,The Burger House And Crunchy Fried Chicken
6833	Tea OClock	Tea OClock,Tea O Clock
4654	Nic Asia BankAtm	Nic Asia BankAtm,Nic Asia Bank Atm
2534	Himalayan Bank	Himalayan Bank,Himalaya Bank
3007	Kamala Khaja Ghar	Kamala Khaja Ghar,Kamal Khaja Ghar
3050	Kareshwar Temple	Kareshwar Temple,Kedareshwar Temple
4774	Nuwakote Khaja Ghar	Nuwakote Khaja Ghar,Nuwakot Khaja Ghar
6668	Suraj Khaja Ghar	Suraj Khaja Ghar,Surya Khaja Ghar
6600	Sumeru Saving And Credit CoOperative Ltd.	Sumeru Saving And Credit CoOperative Ltd.,Sumeru Saving And Credit Cooperative Ltd

Show 25 per page

- check row and column

```
filtered_df.shape
```

```
(471, 2)
```

- Replace Similar Value with Same in `amenity_name`

```
# Apply a lambda function to replace values in 'amenity_name' column
df['amenity_name'] = df['amenity_name'].apply(lambda x:
next((row['amenity'] for _, row in filtered_df.iterrows() if x in row['amenity_similar_values']), x))

df.sample(10)
```

tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value	data_source
9573	node	10053465370	1	True	2022-09-26 11:14:38+00:00	0	27.6804112,85.3656384	2	cafe	15 Hours Cafe And Food Land	0	0	-
1464	node	2185607000	4	True	2022-05-24 06:42:31+00:00	0	27.6843781,85.3456109	10	college	Cite	0	0	-
7242	node	9926053073	3	True	2023-06-15 09:50:37+00:00	0	27.7081555,85.3172143	6	social_facility	Hajis Food Hub	0	0	-
9991	node	10121417239	2	True	2022-11-01 09:13:58+00:00	0	27.6841526,85.3476188	2	parcel_locker	Magic Trade Internatinal Cargo And Courier Ser...	0	0	-
3143	node	4476652593	4	True	2023-06-15 09:50:37+00:00	0	27.7130178,85.2894074	4	post_office	Marshyangdi Health Club	0	0	-
6231	node	9450088717	1	True	2022-01-26 13:06:50+00:00	0	27.7141402,85.3427588	2	restaurant	Ghalcha Restro	0	0	-
9771	node	10108155304	2	True	2023-06-15 08:37:30+00:00	0	27.688672,85.3711101	2	restaurant	Madrix	0	0	-
9998	node	10121417256	1	True	2022-10-22 16:27:18+00:00	0	27.6845515,85.3478646	3	restaurant	Dlight Cafe And Fast Food	0	0	-
10419	node	10629351105	1	True	2023-02-09 07:59:47+00:00	0	27.6959672,85.3769527	3	bank	Ime International Money Express	0	0	-
528	node	1933467831	4	True	2023-06-08 10:05:38+00:00	0	27.6821284,85.3343784	5	place_of_worship	Yantapau Ganesh	0	0	-

Handle Duplicates

Find and assign the duplicates flag which has closest coordinates distances. (i.e. less than 30m) with same name and type.

- find the duplicates and sort it.

> 💡 Note: The first keep value doesn't flag(true) in first duplicate value.

```
# Find duplicate rows based on 'amenity_type' and 'amenity_name'
duplicate_rows = df[df.duplicated(subset=['amenity_type', 'amenity_name'], keep='first')]

# Sort the DataFrame based on columns 'amenity_type' and 'amenity_name'
sorted_duplicate_rows = duplicate_rows.sort_values(by=['amenity_type', 'amenity_name'])

# Display the sorted duplicate rows
sorted_duplicate_rows.head(15)
```

tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value
7293	node	9928590062	1	True	2022-08-03 04:20:19+00:00	0	27.6941437,85.3138641	2	atm	Atm Lounge	0	0
7704	node	9969194287	1	True	2022-08-22 05:47:23+00:00	0	27.6751689,85.3445393	2	atm	Atm Lounge	0	0
753	node	1979178682	6	True	2023-05-26 14:16:34+00:00	0	27.7183118,85.3108415	6	atm	Everest Bank	0	0
6059	node	8699951788	3	True	2023-09-22 15:30:01+00:00	0	27.7173062,85.3265191	6	atm	Everest Bank	0	0
3386	node	4621700406	3	True	2023-06-15 09:50:37+00:00	0	27.693849,85.3145733	4	atm	Everest Bank Atm	0	0
7570	node	9956485672	2	True	2023-05-22 17:19:10+00:00	0	27.6751499,85.3515793	2	atm	Everest Bank Atm	0	0
7230	node	9916725532	2	True	2023-06-14 06:54:37+00:00	0	27.6990074,85.3216193	4	atm	Global Ime Bank Atm	0	0
7283	node	9928566185	3	True	2023-06-15 09:50:37+00:00	0	27.6914834,85.3179324	4	atm	Global Ime Bank Atm	0	0
9859	node	10121388560	1	True	2022-10-22 16:08:31+00:00	0	27.6828472,85.3490047	2	atm	Kumari Bank Atm	0	0
9884	node	10121388674	1	True	2022-10-22 16:08:31+00:00	0	27.6859023,85.3437766	2	atm	Kumari Bank Atm	0	0
3898	node	4866503464	2	True	2018-11-07 09:34:21+00:00	0	27.7159015,85.3040068	3	atm	Laxmi Bank Atm	0	0
7305	node	9928590091	1	True	2022-08-03 04:20:19+00:00	0	27.6985619,85.3132144	2	atm	Laxmi Bank Atm	0	0
9858	node	10121388555	1	True	2022-10-22 16:08:31+00:00	0	27.6821001,85.3490936	2	atm	Mega Bank Atm	0	0
6779	node	9529837575	1	True	2022-02-23 12:56:33+00:00	0	27.7035988,85.3103201	2	atm	Mega Bank Nepal Limited	0	0
697	node	1962703814	4	True	2015-07-17 06:44:04+00:00	0	27.7190344,85.3315418	2	atm	Nabil Bank	0	0

- Assign duplicate flag in `sorted_duplicate_row` df

```
from geopy.distance import geodesic

# Function to calculate distance between two coordinates
def calculate_distance(coord1, coord2):
    lat1, lon1 = map(float, coord1.split(','))
    lat2, lon2 = map(float, coord2.split(','))
    return geodesic((lat1, lon1), (lat2, lon2)).meters

# Create a new column 'is_duplicate' based on specified conditions
sorted_duplicate_rows['is_duplicate'] = False
```

```
# Iterate through all pairs of rows
for i in range(len(sorted_duplicate_rows)):
    for j in range(i + 1, len(sorted_duplicate_rows)):  # Start the inner loop from i + 1
        if sorted_duplicate_rows['amenity_type'].iloc[i] == sorted_duplicate_rows['amenity_type'].iloc[j] and sorted_duplicate_rows['ar
            distance = calculate_distance(sorted_duplicate_rows['coordinates'].iloc[i], sorted_duplicate_rows['coordinates'].iloc[j])
            if distance < 30:
                sorted_duplicate_rows.at[i, 'is_duplicate'] = True
                sorted_duplicate_rows.at[j, 'is_duplicate'] = True

#display which have true flag
duplicate_flag=sorted_duplicate_rows[sorted_duplicate_rows['is_duplicate'].isin([True])]
duplicate_flag
```

tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value	data_source	is_duplicate
20	node	9914340254	1	True	2022-07-27 10:56:55+00:00	0	27.6760288,85.3127076	2	atm	Nabil Bank	0	0	-	True
21	node	9914356110	1	True	2022-07-27 11:11:31+00:00	0	27.6760567,85.3129384	2	atm	Nabil Bank	0	0	-	True
51	node	1783307468	1	True	2012-06-11 08:29:36+00:00	0	27.6729825,85.3243515	1	atm	nan	0	0	-	True
53	node	1857063152	2	True	2023-05-16 05:26:07+00:00	0	27.6982381,85.3131777	2	atm	nan	0	0	-	True
54	node	1882000794	1	True	2012-08-25 06:06:06+00:00	0	27.6884303,85.333801	2	atm	nan	0	0	-	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2598	node	9599541318	1	True	2022-03-23 11:34:31+00:00	0	27.7023491,85.3597784	7	toilets	nan	0	0	-	True
2599	node	9599541319	1	True	2022-03-23 11:34:31+00:00	0	27.7021763,85.3597784	7	toilets	nan	0	0	-	True
2600	node	9599541321	1	True	2022-03-23 11:34:31+00:00	0	27.7023503,85.3597167	5	toilets	nan	0	0	-	True
2614	node	4329294292	1	True	2016-07-31 11:32:36+00:00	0	27.6731048,85.3248909	1	waste_basket	nan	0	0	-	True
2618	node	6291349486	1	True	2019-02-20 15:28:56+00:00	0	27.6730827,85.3249822	1	waste_basket	nan	0	0	-	True

409 rows × 14 columns

- Now assign duplicate flag in main `df`

```
df['is_duplicate'] = df['element_id'].isin(duplicate_flag['element_id'])

#Disply row which have true flag
df[df['is_duplicate'].isin([True])].head(10)
```

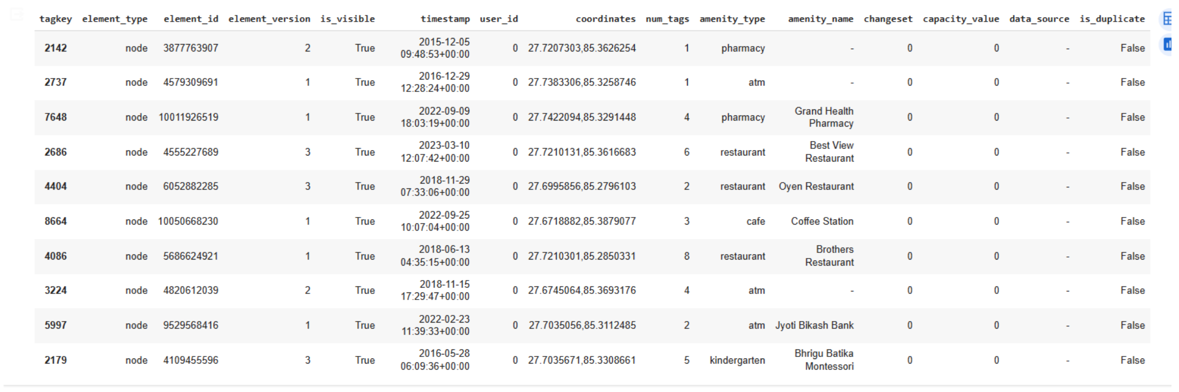
- Remove duplicates row which have true flag.

```
df = df.drop(df[df['is_duplicate'] == True].index).reset_index(drop=True)

#Disply row which have true flag
df[df['is_duplicate'].isin([True])].head(10)
```

tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value	data_source	is_duplicate

- Handle `nan` value in `amenity_name`
    - Replace it with a `-`

```
df['amenity_name'] = df['amenity_name'].replace('nan', '-')
df.sample(10)
```



tagkey	element_type	element_id	element_version	is_visible	timestamp	user_id	coordinates	num_tags	amenity_type	amenity_name	changeset	capacity_value	data_source	is_duplicate
2142	node	3877763907	2	True	2015-12-05 09:48:53+00:00	0	27.7207303,85.3626254	1	pharmacy	-	0	0	-	False
2737	node	4579309691	1	True	2016-12-29 12:28:24+00:00	0	27.7383306,85.3258746	1	atm	-	0	0	-	False
7648	node	10011926519	1	True	2022-09-09 18:03:19+00:00	0	27.7422094,85.3291448	4	pharmacy	Grand Health Pharmacy	0	0	-	False
2686	node	4555227689	3	True	2023-03-10 12:07:42+00:00	0	27.7210131,85.3616683	6	restaurant	Best View Restaurant	0	0	-	False
4404	node	6052882285	3	True	2018-11-29 07:33:06+00:00	0	27.6995856,85.2796103	2	restaurant	Oyen Restaurant	0	0	-	False
8664	node	10050668230	1	True	2022-09-25 10:07:04+00:00	0	27.6718882,85.3879077	3	cafe	Coffee Station	0	0	-	False
4086	node	5686624921	1	True	2018-06-13 04:35:15+00:00	0	27.7210301,85.2850331	8	restaurant	Brothers Restaurant	0	0	-	False
3224	node	4820612039	2	True	2018-11-15 17:29:47+00:00	0	27.6745064,85.3693176	4	atm	-	0	0	-	False
5997	node	9529568416	1	True	2022-02-23 11:39:33+00:00	0	27.7035056,85.3112485	2	atm	Jyoti Bikash Bank	0	0	-	False
2179	node	4109455596	3	True	2016-05-28 06:09:36+00:00	0	27.7035671,85.3308661	5	kindergarten	Bhrigu Batika Montessori	0	0	-	False

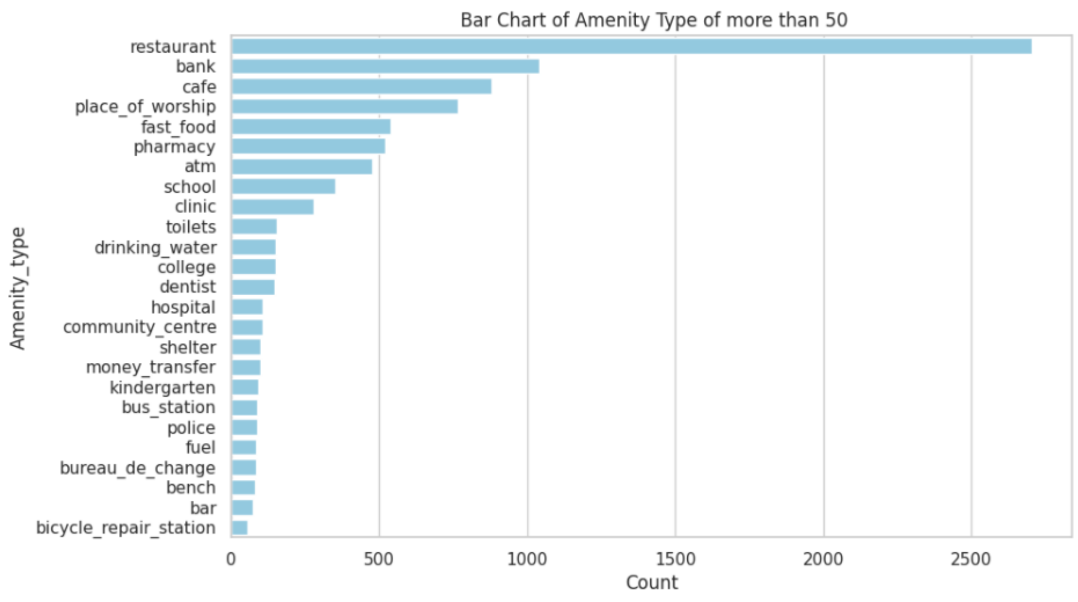
Store in CSV File

```
df.to_csv('Kathmandu_OSM.csv', index=False)
```

3. Create Basic Visualizations

- List the highest Count of `amenity_type` in bar chart.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Get value counts for amenities
amenity_counts = df['amenity_type'].value_counts()

# Filter amenities with counts greater than 50
filtered_amenities = amenity_counts[amenity_counts > 50]

# Set Seaborn style
sns.set_theme(style="whitegrid")

# Create a horizontal bar graph using Seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=filtered_amenities.values, y=filtered_amenities.index, color='skyblue')
plt.title(' Bar Chart of Amenity Type of more than 50')
plt.xlabel('Count')
plt.ylabel('Amenity_type')
plt.show()
```



- Restaurant is Highest. Now, we plot in the map by using folium package.

```
import folium
from folium.plugins import MarkerCluster

# Example types
types = ['restaurant']

# Filter rows where 'amenity_type' is in the specified types and 'coordinates' is not null
filtered_data = df[(df['amenity_type'].isin(types)) & df['coordinates'].notna()]

# Extract latitude and longitude from the 'coordinates' column
coordinates_split = filtered_data['coordinates'].str.split('/', expand=True).astype(float)
filtered_data['latitude'] = coordinates_split[0]
filtered_data['longitude'] = coordinates_split[1]

# Create a map centered around the first location
if not filtered_data.empty:
    map_center = [filtered_data['latitude'].iloc[0], filtered_data['longitude'].iloc[0]]
    map_osm = folium.Map(location=map_center, zoom_start=12)

    # Create a MarkerCluster layer for better visualization if there are multiple points
    marker_cluster = MarkerCluster().add_to(map_osm)

    # Add markers for each location
    for index, row in filtered_data.iterrows():
        folium.Marker(
            location=[row['latitude'], row['longitude']],
        ).add_to(marker_cluster)

else:
    print("No data for specified types with valid coordinates.")
```

💡 Plot on the basis of `amenity_type` values.

- Now display the `map_osm`.

```
#display map
map_osm
```