

# Installing & Importing

```
# First step was to install openpyxl to make it possible to read excel files
!pip install openpyxl==3.1.2
```

Collecting openpyxl==3.1.2

Using cached openpyxl-3.1.2-py2.py3-none-any.whl (249 kB)

Collecting et-xmlfile

Downloading et\_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)

Installing collected packages: et-xmlfile, openpyxl

Successfully installed et-xmlfile-1.1.0 openpyxl-3.1.2

[notice] A new release of pip is available: 23.0.1 -> 24.0

[notice] To update, run: pip install --upgrade pip

```
# Import important libraries
import numpy as np
import pandas as pd
import openpyxl
```

```
df = pd.read_excel("UPDATED_BMMS_overview.xlsx")

df
```

	road object	km float64	type object	LRPName object	name object	length float64	condition object	structureNr int64
	N1 ..... 3.9%	0.004 - 522.718	Box Culvert ..... 44%	LRP003a ..... 2.3%				
	N2 ..... 2.7%		RCC Girder ... 19.1%	LRP001a ..... 2.2%				
	706 others ..... 93.4%		12 others ..... 36.9%	1492 others ..... 95.4%				
0	N1	1.8	Box Culvert	LRP001a	.	11.3	A	117861
1	N1	4.925	Box Culvert	LRP004b	.	6.6	A	117862
2	N1	8.976	PC Girder Bridge	LRP008b	Kanch pur Bridge.	394.23	A	119889
3	N1	10.88	Box Culvert	LRP010b	NOYAPARA CULV...	6.3	A	112531
4	N1	10.897	Box Culvert	LRP010c	ADUPUR CULVERT	6.3	A	112532
5	N1	11.296	Box Culvert	LRP011a	NAYABARI KASP...	8.3	A	101110
6	N1	12.239	Box Culvert	LRP012a	KHAS PARA BOX ...	9.3	A	101117
7	N1	12.253	Box Culvert	LRP012b	DAWAN BAG BOX...	6.1	A	101119
8	N1	12.66	PC Girder Bridge	LRP013a	Madanpur Bridge....	27.5	A	119897
9	N1	12.66	PC Girder Bridge	LRP013a	MADAN PUR (R)	26.3	A	109841

20415 rows, showing 10 per page

<< < Page 1 of 2042 > >>

# Find roads that intersect with N1 & N2

```
#Manually give the reference list of the intersections with road N1 and N2. These will be added in intersect_list.

intersect_list = []
reference_list_N2 = [ 'N1', 'N2', 'R201', 'Z1090', 'R202', 'N105', 'R203', 'Z2038', 'R301', 'Z2047', 'R210', 'R310', 'R211', 'Z2033', 'Z
reference_list_N1 = [ 'Z1101', 'R110', 'Z1029', 'Z1052', 'R141', 'Z1031', 'N104', 'Z1070', 'Z1126', 'R111', 'N2', 'R113', 'N105', 'Z1061
reference_list = reference_list_N1 + reference_list_N2

# Filtering DataFrame based on our reference list and finding the maximum value in column chainage.
max_values = df[df['road'].isin(reference_list)].groupby('road')['chainage'].max()

# Loop through the max_values and append the references to intersect_list if the maximum value is greater than 25
for reference, max_value in max_values.items():
    if max_value > 25:
        intersect_list.append(reference)

print("References added to intersect_list if the maximum value is greater than 25:")
print(intersect_list)
```

References added to intersect\_list if the maximum value is greater than 25:  
['N1', 'N102', 'N104', 'N105', 'N2', 'N204', 'N207', 'N208', 'R141', 'R151', 'R170', 'R203', 'R211', 'R220', 'R240', 'R241', 'R301', 'R360', 'Z1005', 'Z1031',

```
#Remove the duplicates from the list.

def remove_duplicates(input_list):
    """
    Removes duplicates from a list.

    Parameters:
    input_list (list): The list from which duplicates will be removed.

    Returns:
    list: A list with duplicates removed.
    """
    # Convert the list to a set to automatically remove duplicates
    unique_list = list(set(input_list))

    return unique_list

unique_list = remove_duplicates(intersect_list)

print(unique_list)
```

['Z1031', 'R170', 'R360', 'Z1044', 'N204', 'R211', 'R141', 'R151', 'Z1048', 'N208', 'N102', 'N1', 'N207', 'Z1034', 'N105', 'Z1402', 'R301', 'R241', 'N104', 'R2

Create df for bridges

```
# Filtering DataFrame based on our unique_list
df_filtered_bridges = df[df['road'].isin(unique_list)]

df_filtered_bridges
```

	road object	km float64	type object	LRPName object	name object	length float64	condition object	structureNr int64
	N1 ..... 28.1%	0.016 - 460.113	Box Culvert ..... 54.3%	LRP034a ..... 0.8%	..... 3.2%	0.8 - 1408.8	A ..... 60.3%	100018 - 121846
	N2 ..... 20%		RCC Girder ... 20.3%	LRP028a ..... 0.8%	BOX CULVERT 0.6%		C ..... 18.9%	
	23 others ..... 51.9%		11 others ..... 25.4%	1083 others ..... 98.4%	2484 others ..... 96.1%		2 others ..... 20.8%	
198..	Z1402	7.038	RCC Girder Bridge	LRP006c	Arong Bazar Bridge	8.85	D	119844
198..	Z1402	7.038	RCC Girder Bridge	LRP006c	ARONG BAZAR B...	8	D	103857
198..	Z1402	8.627	RCC Girder Bridge	LRP008a	BAROTHALI RCC ...	9.2	D	103879
198..	Z1402	9.756	RCC Girder Bridge	LRP009b	WAPDA bridge	11.5	D	119846
198..	Z1402	9.756	RCC Girder Bridge	LRP009b	OBDA BRIDGA	11	D	103874
198...	Z1402	10.519	RCC Girder Bridge	LRP010a	NABAKALASH	11.75	D	103877
198...	Z1402	16.959	Slab Culvert	LRP016c	NORTH NAGDA S...	3	D	103870
198...	Z1402	18.38	Slab Culvert	LRP018a	SOUTH GILATALI ...	3.5	D	103827
198...	Z1402	20.966	Slab Culvert	LRP020d	PACH GORIA SLA...	4.2	D	103853
198...	Z1402	27.582	Box Culvert	LRP028a	KANACHOR BOX ...	1.2	D	103843

2801 rows, showing 10 per page << < Page 280 of 281 >> >

# Sorting DataFrame based on road (primary) and chainage (secondary)  
df\_sorted\_bridges = df\_filtered\_bridges.sort\_values(by=[ 'road', 'chainage' ])

df\_sorted\_bridges

	road object	km float64	type object	LRPName object	name object	length float64	condition object	structureNr int64
	N1 ..... 28.1% N2 ..... 20% 23 others ..... 51.9%	0.016 - 460.113 	Box Culvert ... 54.3% RCC Girder ... 20.3% 11 others ..... 25.4%	LRP034a ..... 0.8% LRP028a ..... 0.8% 1083 others ..... 98.4%	. ..... 3.2% BOX CULVERT ..... 0.6% 2484 others ..... 96.1%	0.8 - 1408.8 	A ..... 60.3% C ..... 18.9% 2 others ..... 20.8%	100018 - 121846 
7	N1	12.253	Box Culvert	LRP012b	DAWAN BAG BOX...	6.1	A	101119
8	N1	12.66	PC Girder Bridge	LRP013a	Madanpur Bridge....	27.5	A	119897
9	N1	12.66	PC Girder Bridge	LRP013a	MADAN PUR (R)	26.3	A	109841
10	N1	12.66	PC Girder Bridge	LRP013a	MADANPUR BRID...	26.3	A	109838
11	N1	12.688	PC Girder Bridge	LRP013b	Madanpur Bridge...	27.5	A	119900
12	N1	13.574	Box Culvert	LRP014a	KAWTALA BOX C...	11.9	A	109794
145...	N1	15.465	Box Culvert	LRP016a	MALIBAG BOX C...	5.55	B	109800
127...	N1	17.134	RCC Girder Bridge	LRP017b	Langalbandh Brid...	159.52	C	119909
145...	N1	17.134	PC Girder Bridge	LRP017b	LANGOLBANDO ...	159.5	B	109808
13	N1	17.222	PC Girder Bridge	LRP018b	Darikandi Bridge...	20.45	A	119926

2801 rows, showing 10 per page << < Page 2 of 281 > >>

# Add a new column 'model\_type' to the DataFrame  
df\_sorted\_bridges['model\_type'] = 'bridge'

df\_sorted\_bridges

	road object	km float64	type object	LRPName object	name object	length float64	condition object	structureNr int64
	N1 ..... 28.1% N2 ..... 20% 23 others ..... 51.9%	0.016 - 460.113 	Box Culvert ... 54.3% RCC Girder ... 20.3% 11 others ..... 25.4%	LRP034a ..... 0.8% LRP028a ..... 0.8% 1083 others ..... 98.4%	. ..... 3.2% BOX CULVERT ..... 0.6% 2484 others ..... 96.1%	0.8 - 1408.8 	A ..... 60.3% C ..... 18.9% 2 others ..... 20.8%	100018 - 121846 
0	N1	1.8	Box Culvert	LRP001a	.	11.3	A	117861
1	N1	4.925	Box Culvert	LRP004b	.	6.6	A	117862
2	N1	8.976	PC Girder Bridge	LRP008b	Kanch pur Bridge.	394.23	A	119889
127...	N1	8.976	PC Girder Bridge	LRP008b	KANCHPUR PC G...	397	C	101102
145...	N1	10.543	Box Culvert	LRP010a	KATCHPUR BOX ...	8	B	101106
3	N1	10.88	Box Culvert	LRP010b	NOYAPARA CULV...	6.3	A	112531
4	N1	10.897	Box Culvert	LRP010c	ADUPUR CULVERT	6.3	A	112532
5	N1	11.296	Box Culvert	LRP011a	NAYABARI KASP...	8.3	A	101110
145...	N1	11.808	Box Culvert	LRP011c	NAYABARI BOX C...	10.6	B	101114
6	N1	12.239	Box Culvert	LRP012a	KHAS PARA BOX ...	9.3	A	101117

2801 rows, showing 10 per page << < Page 1 of 281 > >>

Create df for roads

roads = pd.read\_csv('\_roads3.csv')

roads

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 2.6%							
	N5 ..... 2.4%							
	845 others ..... 95%							
513...	Z8943	6.706	LRP006a	22.428666	90.7884163	nan	SideRoad,Left	Z8948 to Khaser...
513...	Z8943	6.712	LRP006b	22.4287216	90.7884163	BS	Bridge	Start of Bridge
513...	Z8943	6.742	LRP006c	22.4288049	90.7881386	BE	Bridge	End of Bridge
513...	Z8943	7	LRP007	22.4294993	90.7857219	nan	KmPost	Km Post Missing
513...	Z8943	8	LRP008	22.4301104	90.7763327	nan	KmPost	Km Post Missing
513...	Z8943	8.133	LRP008a	22.4302493	90.7748882	BS	Bridge	Start of Bridge
513...	Z8943	8.151	LRP008b	22.4302493	90.7748604	BE	Bridge	End of Bridge
513...	Z8943	8.8	LRPE	22.430166	90.768916	nan	Others	End at Kunjerhat ...

51348 rows, showing 10 per page

<< < Page 5135 of 5135 > >>

↓

# Filtering DataFrame based on unique\_list

filtered\_roads = roads[roads['road'].isin(unique\_list)]

filtered\_roads

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 24.8%	0.0 - 462.254	LRPS ..... 0.5%	20.8629167 - 25.18...	90.3582222 - 92.3...	BS ..... 10.2%	Culvert ..... 36.6%	Box culvert ..... 20.8%
	N2 ..... 16.4%		LRP001 ..... 0.5%			5 others ..... 10.3%	KmPost ..... 35.2%	Box Culvert ..... 14.7%
	23 others ..... 58.8%		1824 others ..... 99.1%			Missing ..... 79.6%	26 others ..... 28.2%	1261 others ..... 64.5%
30	N1	12.672	LRP013a	23.6854997	90.5512778	nan	Culvert	Box culvert
31	N1	13.524	LRP014	23.678166	90.5556111	nan	KmPost	Km post missing
32	N1	14.524	LRP015	23.670916	90.5597778	nan	KmPost	Km post missing
33	N1	14.563	LRP015a	23.6705827	90.5599722	nan	Culvert	Box culvert
34	N1	15.524	LRP016	23.6628333	90.5641111	nan	KmPost	Km post missing
35	N1	15.935	LRP016a	23.6604167	90.5671104	nan	SideRoad,Right	Road to Minanbar...
36	N1	16.242	LRP016b	23.6591111	90.5697216	BS	Bridge	Langalbandhu bri...
37	N1	16.402	LRP016c	23.6583056	90.5709993	BE	Bridge	Bridge end
38	N1	16.524	LRP017	23.6575	90.5719716	nan	KmPost	Km post missing
39	N1	16.831	LRP017a	23.6559722	90.5743327	BS	Bridge	Bridge start

5402 rows, showing 10 per page

<< < Page 4 of 541 > >>

↓

```
# Add a new column 'model_type' to the DataFrame
filtered_roads['model_type'] = 'link'

df_sorted_roads = filtered_roads.copy()
df_sorted_roads
```

/tmp/ipykernel\_86/195148864.py:3: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
filtered_roads['model_type'] = 'link'
```

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 24.8% N2 ..... 16.4% 23 others ..... 58.8%	0.0 - 462.254 	LRPS ..... 0.5% LRP001 ..... 0.5% 1824 others ..... 99.1%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	BS ..... 10.2% 5 others ..... 10.3% Missing ..... 79.6%	Culvert ..... 36.6% KmPost ..... 35.2% 26 others ..... 28.2%	Box culvert ..... 20.8% Box Culvert ..... 14.7% 1261 others ..... 64.5%
10	N1	6	LRP006	23.6959163	90.4981941	nan	KmPost	Km post missing
11	N1	7	LRP007	23.697333	90.50775	nan	KmPost	Km post missing
12	N1	7.181	LRP007a	23.6979163	90.5092778	nan	CrossRoad	R110,Left to Demr...
13	N1	8	LRP008	23.7020556	90.5157222	nan	KmPost	Km post missing
14	N1	8.011	LRP008a	23.7021111	90.5157778	BS	Bridge	Kachpur bridge
15	N1	8.429	LRP008b	23.7045833	90.5188327	BE	Bridge	Bridge end
16	N1	8.503	LRP009	23.7050278	90.5193327	nan	KmPost	Chittagong 251 k...
17	N1	8.763	LRP009a	23.7060833	90.5215271	nan	SideRoad,Left	Road to Sylhet (N...
18	N1	9.503	LRP010	23.7025278	90.5273882	nan	KmPost	Km post missing
19	N1	9.615	LRP010a	23.7015278	90.5281938	nan	Culvert	Box culvert

5402 rows, showing 10 per page

<< < Page 2 of 541 > >>

↓

## Merge dataframes

```
#####
#Check if all the roads are in df_sorted_roads

# List of values to check
check_list = unique_list

# Check if all values in the list are present in the column 'road' of the DataFrame
all_present = df_sorted_roads['road'].isin(check_list).all()

if all_present:
    print("All values are present in the column.")
else:
    print("Not all values are present in the column.")
```

All values are present in the column.

```
#####
#Check if all the roads are in df_sorted_bridges

# List of values to check
check_list = unique_list

# Check if all values in the list are present in the column 'road' of the DataFrame
all_present = df_sorted_bridges['road'].isin(check_list).all()

if all_present:
    print("All values are present in the column.")
else:
    print("Not all values are present in the column.")
```

All values are present in the column.

```
# Concatenate the two DataFrames vertically
concatenated_df = pd.concat([df_sorted_roads, df_sorted_bridges])

# Sort the concatenated DataFrame based on road (primary) and chainage (secondary)
sorted_df = concatenated_df.sort_values(by=['road', 'chainage'])

# Reset the index of the DataFrame
sorted_df.reset_index(drop=True, inplace=True)

sorted_df
```

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9% N2 ..... 17.6% 23 others ..... 56.5%	0.0 - 462.254 	LRPS ..... 0.3% 1825 others ..... 65.5% Missing ..... 34.1%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	BS ..... 6.7% 5 others ..... 6.8% Missing ..... 86.5%	Culvert ..... 24.1% KmPost ..... 23.2% 39 others ..... 52.7%	Box culvert ..... 13.7% Box Culvert ..... 9.7% 3745 others ..... 76.6%
0	N1	0	LRPS	23.7060278	90.443333	nan	Others	Start of Road afte...
1	N1	0.814	LRPSa	23.7029167	90.4504167	nan	Culvert	Box Culvert
2	N1	0.822	LRPSb	23.7027778	90.4504722	nan	CrossRoad	Intersection with ...
3	N1	1	LRP001	23.7021389	90.4519722	nan	KmPost	Km post missing
4	N1	1.8	nan	23.69873866	90.45886108	nan	Box Culvert	.
5	N1	2	LRP002	23.6978886	90.4605833	nan	KmPost	Km post missing
6	N1	2.13	LRP002a	23.6973608	90.4616667	nan	Culvert	Box culvert
7	N1	3	LRP003	23.693833	90.4691382	nan	KmPost	Km post missing
8	N1	4	LRP004	23.6936108	90.4787771	nan	KmPost	Km post missing
9	N1	4.175	LRP004a	23.6938052	90.4805271	nan	SideRoad,Right	Road to Narayan...

8203 rows, showing 10 per page

<< < Page 1 of 821 > >>

Download

Place sourcesinks and intersections

```
#####
#First the sourcesinks have to be placed so that the intersections in a next step will overwrite certain sourcesinks.

df1 = sorted_df.copy()
# Initialize a variable to keep track of the previous road value
prev_road = None

# Iterate over each row in the DataFrame
for index, row in df1.iterrows():
    # Check if the current road value is different from the previous one
    if row['road'] != prev_road:
        # Update the 'model_type' column for the previous row (if it exists)
        if prev_road is not None:
            df1.at[prev_index, 'model_type'] = 'sourcesink'

        # Update the 'model_type' column for the current row
        df1.at[index, 'model_type'] = 'sourcesink'

    # Update the previous road value and index for the next iteration
    prev_road = row['road']
    prev_index = index

# Update the 'model_type' column for the last row (if the DataFrame is not empty)
if prev_road is not None:
    df1.at[prev_index, 'model_type'] = 'sourcesink'

df1
```

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9% N2 ..... 17.6% 23 others ..... 56.5%	0.0 - 462.254 	LRPS ..... 0.3% 1825 others ..... 65.5% Missing ..... 34.1%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	BS ..... 6.7% 5 others ..... 6.8% Missing ..... 86.5%	Culvert ..... 24.1% KmPost ..... 23.2% 39 others ..... 52.7%	Box culvert ..... 13.7% Box Culvert ..... 9.7% 3745 others ..... 76.6%
8010	Z1402	1.007	LRP001	23.2679438	90.6941941	nan	KmPost	Inf missing
8011	Z1402	1.389	nan	23.27117321	90.69313574	nan	Baily with Steel D...	Kalivangi Bridge
8012	Z1402	1.389	nan	23.27117321	90.69313574	nan	Baily with Steel D...	KALIVANGTI BAIL...
8013	Z1402	1.398	LRP001a	23.2712493	90.6931108	BS	Bridge	Bridge start
8014	Z1402	1.422	LRP001b	23.2714438	90.6931386	BE	Bridge	Bridge end
8015	Z1402	2.022	LRP002	23.2768049	90.6937219	nan	KmPost	Inf missing
8016	Z1402	2.152	nan	23.27795277	90.69374583	nan	RCC Girder Bridge	MOIDAN KHOLA
8017	Z1402	2.173	LRP002a	23.2781382	90.6937497	nan	Culvert	Box Culvert
8018	Z1402	2.877	nan	23.28431818	90.69487321	nan	RCC Girder Bridge	NORTH RAULDIAH
8019	Z1402	2.904	LRP002b	23.2845552	90.6949163	nan	Culvert	Box Culvert

8203 rows, showing 10 per page

<< < Page 802 of 821 > >>

⬇

```
from math import radians, sin, cos, sqrt, atan2

# Function to calculate the distance between two points (given their lat and lon) using the haversine formula
def haversine(lat1, lon1, lat2, lon2):
    # Convert coordinates from degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # Haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance = 6371 * c # Radius of the Earth in kilometers
    return distance

# Function to check if two points are within a certain distance of each other
def within_distance(lat1, lon1, lat2, lon2, distance_threshold):
    distance = haversine(lat1, lon1, lat2, lon2)
    return distance <= distance_threshold

#Test
haversine(24.0515, 90.8731104, 24.0514722, 90.874166)

0.1072311796712705
```

```
#####

#This cell will not be used in this notebook.
#This cell is to only find intersections with road N1.

#import itertools

#intersect_df = sorted_df
# Set the distance threshold
#distance_threshold = 0.01

# Filter the DataFrame to include only points on road "N1"
#road_N1_df = intersect_df[intersect_df['road'] == 'N1']

# Loop through each pair of points on road "N1" and other roads
#for (index1, row1), (index2, row2) in itertools.product(road_N1_df.iterrows(), intersect_df.iterrows()):
#     # Check if the points are from different roads (excluding comparing with itself)
#     if row1['road'] != row2['road']:
#         # Check if the distance between these two points is within the threshold
#         if within_distance(row1['lat'], row1['lon'], row2['lat'], row2['lon'], distance_threshold):
#             # Update the 'model_type' column for both points to 'intersection'
#             intersect_df.at[index1, 'model_type'] = 'intersection'
#             intersect_df.at[index2, 'model_type'] = 'intersection'

# Display the DataFrame with the updated values in the 'model_type' column
#intersect_df
```

```
#This cell is to find intersections with road N1 and road N2.

df2 = df1.copy()

import itertools

# Set the distance threshold
distance_threshold = 1

# Filter the DataFrame to include only points on road "N1"
road_N1_df = df2[df2['road'] == 'N1']


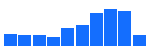
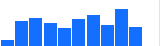
# Filter the DataFrame to include only points on road "N2"
road_N2_df = df2[df2['road'] == 'N2']

# Loop through each pair of points on road "N1" and other roads
for (index1, row1), (index2, row2) in itertools.product(road_N1_df.iterrows(), df2.iterrows()):
    # Check if the points are from different roads (excluding comparing with itself)
    if row1['road'] != row2['road']:
        # Check if the distance between these two points is within the threshold
        if within_distance(row1['lat'], row1['lon'], row2['lat'], row2['lon'], distance_threshold):
            # Update the 'model_type' column for both points to 'intersection'
            df2.at[index1, 'model_type'] = 'intersection'
            df2.at[index2, 'model_type'] = 'intersection'

# Loop through each pair of points on road "N2" and other roads
for (index1, row1), (index2, row2) in itertools.product(road_N2_df.iterrows(), df2.iterrows()):
    # Check if the points are from different roads (excluding comparing with itself)
    if row1['road'] != row2['road']:
        # Check if the distance between these two points is within the threshold
        if within_distance(row1['lat'], row1['lon'], row2['lat'], row2['lon'], distance_threshold):
            # Update the 'model_type' column for both points to 'intersection'
            df2.at[index1, 'model_type'] = 'intersection'
            df2.at[index2, 'model_type'] = 'intersection'

df2
intersect_df = df2.copy()
```

intersect\_df

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9%	0.0 - 462.254	LRPS ..... 0.3%	20.8629167 - 25.18...	90.3582222 - 92.3...	BS ..... 6.7%	Culvert ..... 24.1%	Box culvert ..... 13.7%
	N2 ..... 17.6%		1825 others ..... 65.5%			5 others ..... 6.8%	KmPost ..... 23.2%	Box Culvert ..... 9.7%
	23 others ..... 56.5%		Missing ..... 34.1%			Missing ..... 86.5%	39 others ..... 52.7%	3745 others ..... 76.6%
8200	Z1402	43.117	LRP043	23.5243882	90.7823604	nan	KmPost	Km post missing
8201	Z1402	43.272	LRP043a	23.5257493	90.7822771	nan	Culvert	Box Culvert
8202	Z1402	43.849	LRPE	23.5307493	90.7816104	nan	Others	Meet with N1 at P...

8203 rows, showing 10 per page

<< < Page 821 of 821 > >>

↓



```
# Check for consecutive 'intersection' values in the 'model_type' column to ensure intersections are not doubled due to the threshold.
consecutive_intersections = (intersect_df['model_type'] == 'intersection') & (intersect_df['model_type'].shift(-1) == 'intersection')

# Check if there are any True values in the resulting boolean series
if consecutive_intersections.any():
    print("There are consecutive rows with 'intersection' in the 'model_type' column.")
else:
    print("There are no consecutive rows with 'intersection' in the 'model_type' column.")
```

There are consecutive rows with 'intersection' in the 'model\_type' column.

```
# Check if the current row is the same as the previous row in the 'model_type' column
is_duplicate = intersect_df['model_type'] == intersect_df['model_type'].shift()

# Change the 'model_type' to 'link' if the current row is the same as the previous row
intersect_df.loc[is_duplicate, 'model_type'] = 'link'
```

```
# Check for consecutive 'intersection' values in the 'model_type' column to ensure intersections are not doubled due to the threshold.
consecutive_intersections = (intersect_df['model_type'] == 'intersection') & (intersect_df['model_type'].shift(-1) == 'intersection')

# Check if there are any True values in the resulting boolean series
if consecutive_intersections.any():
    print("There are consecutive rows with 'intersection' in the 'model_type' column.")
else:
    print("There are no consecutive rows with 'intersection' in the 'model_type' column.")
```

There are no consecutive rows with 'intersection' in the 'model\_type' column.

#Check1 intersections

# Filter rows where 'model\_type' is 'intersection'

intersection\_rows = intersect\_df[intersect\_df['model\_type'] == 'intersection']

# Sort the filtered rows based on the values in the 'lat' column

sorted\_intersection\_rows = intersection\_rows.sort\_values(by='lat')

sorted\_intersection\_rows

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N2 ..... 26.9%	0.0 - 339.018	LRPS ..... 25%	21.7080278 - 24.87...	90.5157222 - 92.0...	BS ..... 1.9%	Culvert ..... 30.8%	Box culvert ..... 25%
	N1 ..... 23.1%		27 others ..... 51.9%			Missing ..... 98.1%	Others ..... 25%	Km post mis... 5.8%
	21 others ..... 50%		Missing ..... 23.1%				6 others ..... 44.2%	35 others ..... 69.2%
1261	N1	339.018	LRP343a	21.7080278	92.0800549	nan	Culvert	Box culvert
7107	Z1005	2.267	nan	21.7122292	92.07150284	nan	Slab Culvert	GONAPARA
1251	N1	336.35	LRP340c	21.7299993	92.083833	BS	Bridge	Bridge start
5248	R170	0	LRPS	22.291833	91.9823327	nan	Others	Road start from N...
931	N1	263.578	LRP267a	22.2956941	91.9787493	nan	Culvert	Slab Culvert
5025	R151	0	LRPS	22.8940552	91.5346941	nan	Others	Road start from N...
552	N1	166.427	LRP169a	22.9046667	91.5314438	nan	Culvert	Box culvert
7401	Z1034	1	LRP001	23.0025556	91.4042222	nan	KmPost	Km post missing
2455	N104	0.349	nan	23.0093617	91.39614856	nan	Slab Culvert	Shasaugtachha c...
503	N1	144.67	LRP147b	23.0140833	91.3759716	nan	Culvert	Box culvert

52 rows, showing 10 per page

<< < Page 1 of 6 > >>

↓

```
#Set coordinates of intersections as exactly the same.
mod2_intersect_df = intersect_df.copy()
# Loop through each row in the DataFrame where model_type is 'intersection'
for index1, row1 in mod2_intersect_df[mod2_intersect_df['model_type'] == 'intersection'].iterrows():
    for index2, row2 in mod2_intersect_df[mod2_intersect_df['model_type'] == 'intersection'].iterrows():
        # Check if the distance between these two points is within the threshold
        if index1 != index2 and within_distance(row1['lat'], row1['lon'], row2['lat'], row2['lon'], 3*distance_threshold):
            # If an intersection is detected, copy lon and lat from one road to the other
            mod2_intersect_df.at[index1, 'lat'] = row2['lat']
            mod2_intersect_df.at[index1, 'lon'] = row2['lon']
            mod2_intersect_df.at[index2, 'lat'] = row1['lat']
            mod2_intersect_df.at[index2, 'lon'] = row1['lon']

# Display the DataFrame with the updated values in the 'lat' and 'lon' columns
mod2_intersect_df
```

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9% N2 ..... 17.6% 23 others ..... 56.5%	0.0 - 462.254 	LRPS ..... 0.3% 1825 others ..... 65.5% Missing ..... 34.1%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	BS ..... 6.7% 5 others ..... 6.8% Missing ..... 86.5%	Culvert ..... 24.1% KmPost ..... 23.2% 39 others ..... 52.7%	Box culvert ..... 13.7% Box Culvert ..... 9.7% 3745 others ..... 76.6%
20	N1	8.976	nan	23.70505989	90.52321415	nan	PC Girder Bridge	Kanch pur Bridge.
21	N1	8.976	nan	23.70505989	90.52321415	nan	PC Girder Bridge	KANCHPUR PC G...
22	N1	9.503	LRP010	23.7025278	90.5273882	nan	KmPost	Km post missing
23	N1	9.615	LRP010a	23.7015278	90.5281938	nan	Culvert	Box culvert
24	N1	9.963	LRP010b	23.6998608	90.5307216	nan	Culvert	Box culvert
25	N1	10.377	LRP010c	23.6974163	90.5337774	nan	Culvert	Box culvert
26	N1	10.503	LRP011	23.6966386	90.5348052	nan	KmPost	Km post missing
27	N1	10.543	nan	23.69640009	90.53509898	nan	Box Culvert	KATCHPUR BOX ...
28	N1	10.88	nan	23.69439061	90.53757408	nan	Box Culvert	NOYAPARA CULV...
29	N1	10.885	LRP011a	23.6943608	90.5376108	nan	Culvert	Box culvert

8203 rows, showing 10 per page

<< < Page 3 of 821 > >>

Download

#Check2 intersections

# Filter rows where 'model\_type' is 'intersection'

intersection\_rows = mod2\_intersect\_df[mod2\_intersect\_df['model\_type'] == 'intersection']

# Sort the filtered rows based on the values in the 'lat' column

sorted\_intersection\_rows2 = intersection\_rows.sort\_values(by='lat')

sorted\_intersection\_rows2

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N2 ..... 26.9% N1 ..... 23.1% 21 others ..... 50%	0.0 - 339.018 	LRPS ..... 25% 27 others ..... 51.9% Missing ..... 23.1%	21.7122292 - 24.86... 	90.5214438 - 92.0... 	BS ..... 1.9% Missing ..... 98.1%	Culvert ..... 30.8% Others ..... 25% 6 others ..... 44.2%	Box culvert ..... 25% Km post mis... 5.8% 35 others ..... 69.2%
1251	N1	336.35	LRP340c	21.7122292	92.07150284	BS	Bridge	Bridge start
1261	N1	339.018	LRP343a	21.7122292	92.07150284	nan	Culvert	Box culvert
7107	Z1005	2.267	nan	21.7299993	92.083833	nan	Slab Culvert	GONAPARA
5248	R170	0	LRPS	22.291833	91.9823327	nan	Others	Road start from N...
931	N1	263.578	LRP267a	22.291833	91.9823327	nan	Culvert	Slab Culvert
552	N1	166.427	LRP169a	22.8940552	91.5346941	nan	Culvert	Box culvert
5025	R151	0	LRPS	22.8940552	91.5346941	nan	Others	Road start from N...
7401	Z1034	1	LRP001	23.0025556	91.4042222	nan	KmPost	Km post missing
2455	N104	0.349	nan	23.0025556	91.4042222	nan	Slab Culvert	Shasaugtachha c...
503	N1	144.67	LRP147b	23.0025556	91.4042222	nan	Culvert	Box culvert
7267	Z1031	0	LRPS	23.0614722	91.3616667	nan	Others	Start from Rahma...
484	N1	137.856	nan	23.0614722	91.3616667	nan	Box Culvert	DUTHOSHA BOX ...
8002	Z1048	30.48	nan	23.14890875	91.3193644	nan	Box Culvert	.
441	N1	127.169	LRP129a	23.14890875	91.3193644	nan	Culvert	Box culvert
294	N1	81.801	LRP083a	23.4789716	91.1181938	nan	Culvert	Box culvert
2126	N102	0	LRPS	23.4789716	91.1181938	nan	Others	Start of road fro...
7500	Z1042	0	LRPS	23.5126389	90.871416	nan	Others	Start of Road fro...
232	N1	55.269	LRP056	23.5126389	90.871416	nan	KmPost	Km post Missing
206	N1	44.454	LRP045a	23.5231153	90.78218163	nan	Culvert	Box culvert
7674	Z1044	0	LRPS	23.5231153	90.78218163	nan	Others	Start of Road Fro...
8199	Z1402	42.97	nan	23.5304993	90.7731104	nan	Box Culvert	ISAPUR BOX CUL...
15	N1	8	LRP008	23.6904163	90.5466108	nan	KmPost	Km post missing
2785	N2	0	LRPS	23.69236015	90.54091839	nan	Others	Road Start from ...
31	N1	11.296	nan	23.7059167	90.5214438	nan	Box Culvert	NAYABARI KASP...
2622	N105	0	LRPS	23.7059167	90.5214438	nan	Others	Starts of road fro...
5590	R203	0	LRPS	23.7805549	90.5700549	nan	Others	Road start from N...
2802	N2	10.196	LRP011	23.7856108	90.5705549	nan	KmPost	Km post broken
2657	N105	12.257	LRP012	23.7856108	90.5705549	nan	KmPost	Bangabondhu Bri...
6677	R301	32.12	LRP032	23.9019167	90.6642778	nan	KmPost	Km post missing
2869	N2	29.316	LRP030a	23.9019167	90.6642778	nan	Culvert	Box culvert
2926	N2	42.096	LRP042c	23.980166	90.7334719	nan	Culvert	Box culvert
5911	R211	0	LRPS	23.980166	90.7334719	nan	Others	Start of road fro...
6055	R220	0	LRPS	24.0421108	91.1139722	nan	Others	Road start from N...
2450	N102	81.714	LRP082	24.0576389	91.1150556	nan	KmPost	Madhebpur 23 k...
3135	N2	85.098	LRP085b	24.0576389	91.1150556	nan	Culvert	Box culvert
3066	N2	69.706	LRP070c	24.0623889	90.9799993	nan	Culvert	Box culvert
7091	R360	115.784	LRP115b	24.0623889	90.9799993	nan	Culvert	Box Culvert
4231	N204	0	LRPS	24.1479163	91.3466108	nan	Others	Road start from N...
3277	N2	115.81	nan	24.1479163	91.3466108	nan	Box Culvert	WORLDKOAL CUL...
6225	R240	0	LRPS	24.2615	91.482416	nan	Others	Road start from a...
6230	R240	2.61	LRP002a	24.2676104	91.4768882	nan	RailRoadCrossing	Rail Road Crossing
3430	N2	139.773	nan	24.2768882	91.45525	nan	Box Culvert	BIRMACCHAR CUL...
4353	N204	33.535	LRP035a	24.2768882	91.45525	nan	Culvert	Box culvert
3418	N2	137.262	nan	24.2768882	91.45525	nan	Box Culvert	BORCHAR
3455	N2	145.041	LRP146	24.2892219	91.5034444	nan	KmPost	Sylhet 86 km
6423	R240	48.44	LRP048c	24.5909163	91.5921108	nan	Culvert	Box Culvert
3754	N2	180.655	LRP181c	24.5909163	91.5921108	nan	Culvert	Box culvert
3785	N2	185.04	nan	24.60975831	91.6253431	nan	Box Culvert	JALALPUR
4602	N207	67.56	nan	24.62262803	91.67809144	nan	Box Culvert	MIRZAPUR BOX ...

3811	N2	189.984	nan	24.62262803	91.67809144	nan	PC Girder Bridge	GRAM SHERPUR ...
------	----	---------	-----	-------------	-------------	-----	------------------	------------------

52 rows, showing 50 per page

<< < Page 1 of 2 > >>

#Check3 intersections

# Filter rows where 'model\_type' is 'intersection'

intersection\_rows = mod2\_intersect\_df[mod2\_intersect\_df['model\_type'] == 'intersection']

# Sort the filtered rows based on the values in the 'lat' column

sorted\_intersection\_rows3 = intersection\_rows.sort\_values(by='lat')

sorted\_intersection\_rows3

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N2 ..... 26.9% N1 ..... 23.1% 21 others ..... 50%	0.0 - 339.018 	LRPS ..... 25% 27 others ..... 51.9% Missing ..... 23.1%	21.7122292 - 24.86... 	90.5214438 - 92.0... 	BS ..... 1.9% Missing ..... 98.1%	Culvert ..... 30.8% Others ..... 25% 6 others ..... 44.2%	Box culvert ..... 25% Km post mis... 5.8% 35 others ..... 69.2%
1251	N1	336.35	LRP340c	21.7122292	92.07150284	BS	Bridge	Bridge start
1261	N1	339.018	LRP343a	21.7122292	92.07150284	nan	Culvert	Box culvert
7107	Z1005	2.267	nan	21.7299993	92.083833	nan	Slab Culvert	GONAPARA
5248	R170	0	LRPS	22.291833	91.9823327	nan	Others	Road start from N...
931	N1	263.578	LRP267a	22.291833	91.9823327	nan	Culvert	Slab Culvert
552	N1	166.427	LRP169a	22.8940552	91.5346941	nan	Culvert	Box culvert
5025	R151	0	LRPS	22.8940552	91.5346941	nan	Others	Road start from N...
7401	Z1034	1	LRP001	23.0025556	91.4042222	nan	KmPost	Km post missing
2455	N104	0.349	nan	23.0025556	91.4042222	nan	Slab Culvert	Shasaugtachha c...
503	N1	144.67	LRP147b	23.0025556	91.4042222	nan	Culvert	Box culvert

52 rows, showing 10 per page

<< < Page 1 of 6 > >>

mod3\_intersect\_df = mod2\_intersect\_df.copy()

# Determine the starting number for the ids

start\_id = 1000000

# Add the 'id' column to the DataFrame

mod3\_intersect\_df['id'] = range(start\_id, start\_id + len(mod3\_intersect\_df))

mod3\_intersect\_df

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9% N2 ..... 17.6% 23 others ..... 56.5%	0.0 - 462.254 	LRPS ..... 0.3% 1825 others ..... 65.5% Missing ..... 34.1%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	BS ..... 6.7% 5 others ..... 6.8% Missing ..... 86.5%	Culvert ..... 24.1% KmPost ..... 23.2% 39 others ..... 52.7%	Box culvert ..... 13.7% Box Culvert ..... 9.7% 3745 others ..... 76.6%
30	N1	10.897	nan	23.69430239	90.53770737	nan	Box Culvert	ADUPUR CULVERT
31	N1	11.296	nan	23.7059167	90.5214438	nan	Box Culvert	NAYABARI KASP...
32	N1	11.313	LRP011b	23.6922774	90.5410552	nan	Culvert	Box culvert
33	N1	11.497	LRP012	23.6918052	90.5426108	nan	KmPost	Chittagong 248 k...
34	N1	11.745	LRP012a	23.6911108	90.5448886	BS	Bridge	Bridge start
35	N1	11.771	LRP012b	23.6910552	90.5451386	BE	Bridge	Bridge end
36	N1	11.808	nan	23.69091193	90.5454625	nan	Box Culvert	NAYABARI BOX C...
37	N1	11.936	LRP012c	23.6904163	90.546583	nan	SideRoad,Right	Right to Syedpur ...
38	N1	12.239	nan	23.68841233	90.54855853	nan	Box Culvert	KHAS PARA BOX ...
39	N1	12.253	nan	23.68831973	90.54864981	nan	Box Culvert	DAWAN BAG BOX...

8203 rows, showing 10 per page

<< < Page 4 of 821 > >>

# Reset the indexes of the DataFrame  
mod3\_intersect\_df.reset\_index(drop=True, inplace=True)

mod3\_intersect\_df

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9%	0.0 - 462.254	LRPS ..... 0.3%	20.8629167 - 25.18...	90.3582222 - 92.3...	BS ..... 6.7%	Culvert ..... 24.1%	Box culvert ..... 13.7%
	N2 ..... 17.6%		1825 others ..... 65.5%			5 others ..... 6.8%	KmPost ..... 23.2%	Box Culvert ..... 9.7%
	23 others ..... 56.5%		Missing ..... 34.1%			Missing ..... 86.5%	39 others ..... 52.7%	3745 others ..... 76.6%
30	N1	10.897	nan	23.69430239	90.53770737	nan	Box Culvert	ADUPUR CULVERT
31	N1	11.296	nan	23.7059167	90.5214438	nan	Box Culvert	NAYABARI KASP...
32	N1	11.313	LRP011b	23.6922774	90.5410552	nan	Culvert	Box culvert
33	N1	11.497	LRP012	23.6918052	90.5426108	nan	KmPost	Chittagong 248 k...
34	N1	11.745	LRP012a	23.6911108	90.5448886	BS	Bridge	Bridge start
35	N1	11.771	LRP012b	23.6910552	90.5451386	BE	Bridge	Bridge end
36	N1	11.808	nan	23.69091193	90.5454625	nan	Box Culvert	NAYABARI BOX C...
37	N1	11.936	LRP012c	23.6904163	90.546583	nan	SideRoad,Right	Right to Syedpur ...
38	N1	12.239	nan	23.68841233	90.54855853	nan	Box Culvert	KHAS PARA BOX ...
39	N1	12.253	nan	23.68831973	90.54864981	nan	Box Culvert	DAWAN BAG BOX...

8203 rows, showing 10 per page

<< < Page 4 of 821 >> >>

↓

#!!!!

#Make the id of the intersections the same.  
mod4\_intersect\_df = mod3\_intersect\_df.copy() # Make a copy of the DataFrame to avoid modifying the original one

# Determine the starting number for the ids  
start\_id = 1000000

# Initialize a dictionary to store mappings of (lat, lon) to id  
id\_map = {}

# Loop through each row in the DataFrame where model\_type is 'intersection'  
for index, row in mod4\_intersect\_df[mod4\_intersect\_df['model\_type'] == 'intersection'].iterrows():  
# Get the current latitude and longitude  
lat = row['lat']  
lon = row['lon']  
  
# Check if this (lat, lon) pair already exists in the id\_map  
if (lat, lon) in id\_map:  
# If it exists, assign the same id to this intersection  
mod4\_intersect\_df.at[index, 'id'] = id\_map[(lat, lon)]  
else:  
# If it doesn't exist, assign the current index + 1000000 as the ID and store the mapping  
mod4\_intersect\_df.at[index, 'id'] = index + start\_id  
id\_map[(lat, lon)] = index + start\_id

mod4\_intersect\_df

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9%	0.0 - 462.254	LRPS ..... 0.3%	20.8629167 - 25.18...	90.3582222 - 92.3...	BS ..... 6.7%	Culvert ..... 24.1%	Box culvert ..... 13.7%
	N2 ..... 17.6%		1825 others ..... 65.5%			5 others ..... 6.8%	KmPost ..... 23.2%	Box Culvert ..... 9.7%
	23 others ..... 56.5%		Missing ..... 34.1%			Missing ..... 86.5%	39 others ..... 52.7%	3745 others ..... 76.6%
7720	Z1044	12.006	nan	23.45598088	90.83782816	nan	Box Culvert	BAYAK CULVERT
7721	Z1044	12.116	LRP012a	23.4551944	90.8383052	nan	Culvert	Box culvert
7722	Z1044	12.438	nan	23.45264991	90.83932686	nan	Box Culvert	BAYAK CULVERT
7723	Z1044	12.58	LRP012b	23.4515278	90.8397774	nan	Culvert	Box culvert
7724	Z1044	12.879	LRP013	23.4492497	90.8410552	nan	KmPost	Km Post Eastablis...
7725	Z1044	12.999	nan	23.44796027	90.8409592	nan	Box Culvert	BAYAK CULVERT
7726	Z1044	13.122	LRP013a	23.4466386	90.8408608	nan	Culvert	Box culvert
7727	Z1044	13.306	nan	23.44504337	90.84061662	nan	Box Culvert	SORAIL CULVERT
7728	Z1044	13.436	LRP013b	23.4439163	90.8404441	nan	Culvert	Box culvert
7729	Z1044	13.755	nan	23.44107598	90.84106417	nan	RCC Girder Bridge	SACHAR BRIDGE

8203 rows, showing 10 per page

<< < Page 773 of 821 >> >>

↓

```
# Sort the DataFrame based on the 'road' column and then the 'chainage' column
mod4_intersect_df = mod4_intersect_df.sort_values(by=['road', 'chainage'])
```

Calculate length

```
#Calculate length

# Make a copy of the DataFrame to avoid modifying the original DataFrame
mod5_intersect_df = mod4_intersect_df.copy()

# Filter rows where 'model_type' is 'link'
link_rows = mod5_intersect_df[mod5_intersect_df['model_type'] == 'link']

# Calculate the length only for the filtered rows
link_rows['length'] = link_rows.groupby('road')['chainage'].diff()

# Update the 'length' column in the original DataFrame with the calculated lengths for link rows
mod5_intersect_df.update(link_rows)

# Display the DataFrame with the updated 'length' column
mod5_intersect_df
```

/tmp/ipykernel\_86/2615309740.py:11: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
link_rows['length'] = link_rows.groupby('road')['chainage'].diff()
```

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9%	0.0 - 462.254	LRPS ..... 0.3%	20.8629167 - 25.18...	90.3582222 - 92.3...	BS ..... 6.7%	Culvert ..... 24.1%	Box culvert ..... 13.7%
	N2 ..... 17.6%		1825 others ..... 65.5%			5 others ..... 6.8%	KmPost ..... 23.2%	Box Culvert ..... 9.7%
	23 others ..... 56.5%		Missing ..... 34.1%			Missing ..... 86.5%	39 others ..... 52.7%	3745 others ..... 76.6%

7720	Z1044	12.006	nan	23.45598088	90.83782816	nan	Box Culvert	BAYAK CULVERT
7721	Z1044	12.116	LRP012a	23.4551944	90.8383052	nan	Culvert	Box culvert
7722	Z1044	12.438	nan	23.45264991	90.83932686	nan	Box Culvert	BAYAK CULVERT
7723	Z1044	12.58	LRP012b	23.4515278	90.8397774	nan	Culvert	Box culvert
7724	Z1044	12.879	LRP013	23.4492497	90.8410552	nan	KmPost	Km Post Eastablis...
7725	Z1044	12.999	nan	23.44796027	90.8409592	nan	Box Culvert	BAYAK CULVERT
7726	Z1044	13.122	LRP013a	23.4466386	90.8408608	nan	Culvert	Box culvert
7727	Z1044	13.306	nan	23.44504337	90.84061662	nan	Box Culvert	SORAIL CULVERT
7728	Z1044	13.436	LRP013b	23.4439163	90.8404441	nan	Culvert	Box culvert
7729	Z1044	13.755	nan	23.44107598	90.84106417	nan	RCC Girder Bridge	SACHAR BRIDGE

8203 rows, showing 10 per page

<< < Page 773 of 821 > >>

↓

```
#Calculate chainage in km to length in meters.
mod6_intersect_df = mod5_intersect_df

# Multiply values in the 'length' column by 1000 for rows where 'model_type' is 'link'
mod6_intersect_df.loc[mod6_intersect_df['model_type'] == 'link', 'length'] *= 1000
```

```
# Replace NaN values in a specific column with 0
mod6_intersect_df['length'] = mod6_intersect_df['length'].fillna(0)
```

Pour into the correct format

# Reset the index to consecutive integers  
mod6\_intersect\_df = mod6\_intersect\_df.reset\_index(drop=True)

mod6\_intersect\_df

	road object	chainage float64	lrp object	lat float64	lon float64	gap object	type object	name object
	N1 ..... 25.9% N2 ..... 17.6% 23 others ..... 56.5%	0.0 - 462.254 	LRPS ..... 0.3% 1825 others ..... 65.5% Missing ..... 34.1%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	BS ..... 6.7% 5 others ..... 6.8% Missing ..... 86.5%	Culvert ..... 24.1% KmPost ..... 23.2% 39 others ..... 52.7%	Box culvert ..... 13.7% Box Culvert ..... 9.7% 3745 others ..... 76.6%
0	N1	0	LRPS	23.7060278	90.443333	nan	Others	Start of Road afte...
1	N1	0.814	LRPSa	23.7029167	90.4504167	nan	Culvert	Box Culvert
2	N1	0.822	LRPSb	23.7027778	90.4504722	nan	CrossRoad	Intersection with ...
3	N1	1	LRP001	23.7021389	90.4519722	nan	KmPost	Km post missing
4	N1	1.8	nan	23.69873866	90.45886108	nan	Box Culvert	.
5	N1	2	LRP002	23.6978886	90.4605833	nan	KmPost	Km post missing
6	N1	2.13	LRP002a	23.6973608	90.4616667	nan	Culvert	Box culvert
7	N1	3	LRP003	23.693833	90.4691382	nan	KmPost	Km post missing
8	N1	4	LRP004	23.6936108	90.4787771	nan	KmPost	Km post missing
9	N1	4.175	LRP004a	23.6938052	90.4805271	nan	SideRoad,Right	Road to Narayan...

8203 rows, showing 10 per page << < Page 1 of 821 > >> [Download](#)

mod7\_intersect\_df = mod6\_intersect\_df.copy()

# Select the desired columns in the desired order  
selected\_columns = ['road', 'id', 'model\_type', 'condition', 'name', 'lat', 'lon', 'length']

# Create a new DataFrame with only the selected columns  
mod7\_intersect\_df = mod7\_intersect\_df[selected\_columns]

# Sort the DataFrame based on the index to preserve the original order  
mod7\_intersect\_df = mod7\_intersect\_df.sort\_index()

# Display the resulting DataFrame  
mod7\_intersect\_df

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
	N1 ..... 25.9% N2 ..... 17.6% 23 others ..... 56.5%	1000000 - 1008202 	link ..... 75% bridge ..... 24.1% 2 others ..... 0.9%	A ..... 20.6% 3 others ..... 13.6% Missing ..... 65.9%	Box culvert ..... 13.7% Box Culvert ..... 9.7% 3745 others ..... 76.6%	20.8629167 - 25.18... 	90.3582222 - 92.3... 	0.0 - 12800.0 
30	N1	1000030	bridge	A	ADUPUR CULVERT	23.69430239	90.53770737	6.3
31	N1	1000031	intersection	A	NAYABARI KASP...	23.7059167	90.5214438	8.3
32	N1	1000032	link	nan	Box culvert	23.6922774	90.5410552	428
33	N1	1000033	link	nan	Chittagong 248 k...	23.6918052	90.5426108	184
34	N1	1000034	link	nan	Bridge start	23.6911108	90.5448886	248
35	N1	1000035	link	nan	Bridge end	23.6910552	90.5451386	26
36	N1	1000036	link	B	NAYABARI BOX C...	23.69091193	90.5454625	37
37	N1	1000037	link	nan	Right to Syedpur ...	23.6904163	90.546583	128
38	N1	1000038	link	A	KHAS PARA BOX ...	23.68841233	90.54855853	303
39	N1	1000039	link	A	DAWAN BAG BOX...	23.68831973	90.54864981	14

8203 rows, showing 10 per page << < Page 4 of 821 > >> [Download](#)

```
#Check4 intersections

# Filter rows where 'model_type' is 'intersection'
intersection_rows = mod7_intersect_df[mod7_intersect_df['model_type'] == 'intersection']

# Sort the filtered rows based on the values in the 'lat' column
sorted_intersection_rows4 = intersection_rows.sort_values(by='lat')

# Display the sorted intersection rows
sorted_intersection_rows4
```

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
	N2 ..... 26.9% N1 ..... 23.1% 21 others ..... 50%	1000015 - 1008199	intersection ..... 100%	A ..... 17.3% 2 others ..... 5.8% Missing ..... 76.9%	Box culvert ..... 25% Km post mis... 5.8% 35 others ..... 69.2%	21.7122292 - 24.86...	90.5214438 - 92.0...	0.0 - 16.7
6230	R240	1006230	intersection	nan	Rail Road Crossing	24.2676104	91.4768882	0
3430	N2	1003418	intersection	A	BIRMACHAR CUL...	24.2768882	91.45525	2.1
4353	N204	1003418	intersection	nan	Box culvert	24.2768882	91.45525	0
3418	N2	1003418	intersection	A	BORCHAR	24.2768882	91.45525	2.6
3455	N2	1003455	intersection	nan	Sylhet 86 km	24.2892219	91.5034444	0
6423	R240	1003754	intersection	nan	Box Culvert	24.5909163	91.5921108	0
3754	N2	1003754	intersection	nan	Box culvert	24.5909163	91.5921108	0
3785	N2	1003785	intersection	A	JALALPUR	24.60975831	91.6253431	2
4602	N207	1003811	intersection	A	MIRZAPUR BOX ...	24.62262803	91.67809144	6
3811	N2	1003811	intersection	A	GRAM SHERPUR ...	24.62262803	91.67809144	16.7

52 rows, showing 10 per page

<< < Page 5 of 6 > >>

↓

```
#Remove intersections that occur once so are actually links.

# Count the occurrences of each value in the 'lat' column
value_counts = sorted_intersection_rows4['lat'].value_counts()

# Filter the index of values that occur only once
indices_to_change = value_counts[value_counts == 1].index

# Change the value of the 'model_type' to 'link' for rows where the 'lat' value occurs only once
sorted_intersection_rows4.loc[sorted_intersection_rows4['lat'].isin(indices_to_change), 'model_type'] = 'link'

sorted_intersection_rows4
```

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
	N2 ..... 26.9% N1 ..... 23.1% 21 others ..... 50%	1000015 - 1008199	intersection ..... 80.8% link ..... 19.2%	A ..... 17.3% 2 others ..... 5.8% Missing ..... 76.9%	Box culvert ..... 25% Km post mis... 5.8% 35 others ..... 69.2%	21.7122292 - 24.86...	90.5214438 - 92.0...	0.0 - 16.7
6230	R240	1006230	link	nan	Rail Road Crossing	24.2676104	91.4768882	0
3430	N2	1003418	intersection	A	BIRMACHAR CUL...	24.2768882	91.45525	2.1
4353	N204	1003418	intersection	nan	Box culvert	24.2768882	91.45525	0
3418	N2	1003418	intersection	A	BORCHAR	24.2768882	91.45525	2.6
3455	N2	1003455	link	nan	Sylhet 86 km	24.2892219	91.5034444	0
6423	R240	1003754	intersection	nan	Box Culvert	24.5909163	91.5921108	0
3754	N2	1003754	intersection	nan	Box culvert	24.5909163	91.5921108	0
3785	N2	1003785	link	A	JALALPUR	24.60975831	91.6253431	2
4602	N207	1003811	intersection	A	MIRZAPUR BOX ...	24.62262803	91.67809144	6
3811	N2	1003811	intersection	A	GRAM SHERPUR ...	24.62262803	91.67809144	16.7

52 rows, showing 10 per page

<< < Page 5 of 6 > >>

↓



```
# Step 1: Filter rows with 'model_type' == 'intersection'
intersection_rows = mod7_intersect_df[mod7_intersect_df['model_type'] == 'intersection']

# Step 2: Count occurrences of each value in 'lat' column
lat_counts = intersection_rows['lat'].value_counts()

# Step 3: Identify rows where 'lat' occurs only once
unique_lat_rows = intersection_rows[intersection_rows['lat'].map(lat_counts) == 1]

# Step 4: Update 'model_type' for identified rows to 'link'
mod7_intersect_df.loc[unique_lat_rows.index, 'model_type'] = 'link'

mod7_intersect_df
```

	<div>road object</div> <div>N1 ..... 25.9%</div> <div>N2 ..... 17.6%</div> <div>23 others ..... 56.5%</div>	<div>id int64</div> <div>1000000 - 1008202</div> <div></div>	<div>model_type object</div> <div>link ..... 75.1%</div> <div>bridge ..... 24.1%</div> <div>2 others ..... 0.8%</div>	<div>condition object</div> <div>A ..... 20.6%</div> <div>3 others ..... 13.6%</div> <div>Missing ..... 65.9%</div>	<div>name object</div> <div>Box culvert ..... 13.7%</div> <div>Box Culvert ..... 9.7%</div> <div>3745 others ..... 76.6%</div>	<div>lat float64</div> <div>20.8629167 - 25.18...</div> <div></div>	<div>lon float64</div> <div>90.3582222 - 92.3...</div> <div></div>	<div>length float64</div> <div>0.0 - 12800.0</div> <div></div>
3500	N2	1003500	link	nan	Sylhet 78 km	24.3508333	91.536083	239
3501	N2	1003501	bridge	B	Chalta Bazar Brid...	24.35139197	91.5364729	58.5
3502	N2	1003502	link	B	CHALTA BAZAR B...	24.35139197	91.5364729	75
3503	N2	1003503	link	A	CHALTA BOX CUL...	24.35335847	91.53784533	264
3504	N2	1003504	link	nan	Bridge start	24.3535	91.5379441	19
3505	N2	1003505	link	nan	Bridge end	24.3538611	91.5384441	66
3506	N2	1003506	link	nan	Box culvert	24.3551111	91.5398608	202
3507	N2	1003507	link	nan	Km post broken	24.3582222	91.5411386	384
3508	N2	1003508	bridge	A	CHALTA BAZAR B...	24.36034858	91.54159598	3
3509	N2	1003509	link	nan	Box culvert	24.363	91.5421663	546

8203 rows, showing 10 per page << < Page 351 of 821 >> >>

```
#Check5 intersections

# Filter rows where 'model_type' is 'intersection'
intersection_rows = mod7_intersect_df[mod7_intersect_df['model_type'] == 'intersection']

# Sort the filtered rows based on the values in the 'lat' column
sorted_intersection_rows5 = intersection_rows.sort_values(by='lat')

# Display the sorted intersection rows
sorted_intersection_rows5
```

	<div>road object</div> <div>N1 ..... 26.2%</div> <div>N2 ..... 26.2%</div> <div>17 others ..... 47.6%</div>	<div>id int64</div> <div>1000031 - 1003965</div> <div></div>	<div>model_type object</div> <div>intersection ..... 100%</div>	<div>condition object</div> <div>A ..... 16.7%</div> <div>B ..... 4.8%</div> <div>Missing ..... 78.6%</div>	<div>name object</div> <div>Box culvert ..... 31%</div> <div>Km post mis... ..... 4.8%</div> <div>26 others ..... 64.3%</div>	<div>lat float64</div> <div>21.7122292 - 24.86...</div> <div></div>	<div>lon float64</div> <div>90.5214438 - 92.0...</div> <div></div>	<div>length float64</div> <div>0.0 - 16.7</div> <div></div>
3965	N2	1003965	intersection	nan	Box culvert	24.8694993	91.8760271	0
4887	N208	1003965	intersection	nan	Sylhet 5 km	24.8694993	91.8760271	0

42 rows, showing 10 per page << < Page 5 of 5 >> >>

# Remove isolated sourcesinks

These sourcesinks are removed because the model indicates that they are not connected with other sourcesinks. Both are listed below, and manual inspection determined which ones should be removed. The first two are extensively discussed, while the last three are removed in abbreviated form.

```
# Get all rows where the value in the column 'id' is x
rows_with_value_x = mod7_intersect_df[mod7_intersect_df['id'] == 1004230
]

rows_with_value_x
#N2

# Get all rows where the value in the column 'id' is x
rows_with_value_x = mod7_intersect_df[mod7_intersect_df['id'] == 1006578
]

rows_with_value_x
#R241
```

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
6578	R241	1006578	sourcesink	nan	Road end with R2...	24.9233604	91.4625833	0

1 row, showing 10 per page<< < Page 1 of 1 >>↓

```
# Define the specific value in the 'road' column
specific_value = 'R241'

# Filter the DataFrame based on the specific value in the 'road' column and 'sourcesink' in 'model_type'
mod7_intersect_df.loc[(mod7_intersect_df['road'] == specific_value) & (mod7_intersect_df['model_type'] == 'sourcesink'), 'model_type'] =

mod7_intersect_df
```

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
	N1 25.9% N2 17.6% 23 others 56.5%	1000000 - 1008202	link 75.1% bridge 24.1% 2 others 0.8%	A 20.6% 3 others 13.6% Missing 65.9%	Box culvert 13.7% Box Culvert 9.7% 3745 others 76.6%	20.8629167 - 25.18...	90.3582222 - 92.3...	0.0 - 12800.0
0	N1	1000000	sourcesink	nan	Start of Road afte...	23.7060278	90.4433333	0
1	N1	1000001	link	nan	Box Culvert	23.7029167	90.4504167	0
2	N1	1000002	link	nan	Intersection with ...	23.7027778	90.4504722	8
3	N1	1000003	link	nan	Km post missing	23.7021389	90.4519722	178
4	N1	1000004	bridge	A	.	23.69873866	90.45886108	11.3
5	N1	1000005	link	nan	Km post missing	23.6978886	90.4605833	1000
6	N1	1000006	link	nan	Box culvert	23.6973608	90.4616667	130
7	N1	1000007	link	nan	Km post missing	23.693833	90.4691382	870
8	N1	1000008	link	nan	Km post missing	23.6936108	90.4787771	1000
9	N1	1000009	link	nan	Road to Narayan...	23.6938052	90.4805271	175

8203 rows, showing 10 per page<< < Page 1 of 821 >>↓

```
# Get all rows where the value in the column 'id' is x
rows_with_value_x = mod7_intersect_df[mod7_intersect_df['id'] == 1007266
]

rows_with_value_x
#Z1005

# Get all rows where the value in the column 'id' is x
rows_with_value_x = mod7_intersect_df[mod7_intersect_df['id'] == 1004605
]

rows_with_value_x
#N208
```

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
4605	N208	1004605	sourcesink	nan	Road Start from ...	24.4713604	91.7655556	0

1 row, showing 10 per page<< < Page 1 of 1 >>↓

```
# Define the specific value in the 'road' column
specific_value = 'Z1005'

# Filter the DataFrame based on the specific value in the 'road' column and 'sourcesink' in 'model_type'
mod7_intersect_df.loc[(mod7_intersect_df['road'] == specific_value) & (mod7_intersect_df['model_type'] == 'sourcesink'), 'model_type'] =

mod7_intersect_df
```

	road object	id int64	model_type object	condition object	name object	lat float64	lon float64	length float64
	N1 ..... 25.9%	1000000 - 1008202	link ..... 75.1%	A ..... 20.6%	Box culvert ..... 13.7%	20.8629167 - 25.18...	90.3582222 - 92.3...	0.0 - 12800.0
	N2 ..... 17.6%		bridge ..... 24.1%	3 others ..... 13.6%	Box Culvert ..... 9.7%			
	23 others ..... 56.5%		2 others ..... 0.8%	Missing ..... 65.9%	3745 others ..... 76.6%			
0	N1	1000000	sourcesink	nan	Start of Road afte...	23.7060278	90.443333	0
1	N1	1000001	link	nan	Box Culvert	23.7029167	90.4504167	0
2	N1	1000002	link	nan	Intersection with ...	23.7027778	90.4504722	8
3	N1	1000003	link	nan	Km post missing	23.7021389	90.4519722	178
4	N1	1000004	bridge	A	.	23.69873866	90.45886108	11.3
5	N1	1000005	link	nan	Km post missing	23.6978886	90.4605833	1000
6	N1	1000006	link	nan	Box culvert	23.6973608	90.4616667	130
7	N1	1000007	link	nan	Km post missing	23.693833	90.4691382	870
8	N1	1000008	link	nan	Km post missing	23.6936108	90.4787771	1000
9	N1	1000009	link	nan	Road to Narayan...	23.6938052	90.4805271	175

8203 rows, showing 10 per page

<< < Page 1 of 821 > >>

↓

```
# Filter rows where 'road' column has value 'x' and 'id' column has value 'y'
filtered_rows = mod7_intersect_df[(mod7_intersect_df['road'] == 'Z1402') & (mod7_intersect_df['id'] == '1008009')]

# Update the 'model_type' column for the filtered rows
filtered_rows['model_type'] = 'link'

# Update the original DataFrame with the modified rows
df.update(filtered_rows)
```

```
# Filter rows where 'road' column has value 'x' and 'id' column has value 'y'
filtered_rows = mod7_intersect_df[(mod7_intersect_df['road'] == 'R203') & (mod7_intersect_df['id'] == '1005910')]

# Update the 'model_type' column for the filtered rows
filtered_rows['model_type'] = 'link'

# Update the original DataFrame with the modified rows
df.update(filtered_rows)
```

```
# Filter rows where 'road' column has value 'x' and 'id' column has value 'y'
filtered_rows = mod7_intersect_df[(mod7_intersect_df['road'] == 'R220') & (mod7_intersect_df['id'] == '1006224')]

# Update the 'model_type' column for the filtered rows
filtered_rows['model_type'] = 'link'

# Update the original DataFrame with the modified rows
df.update(filtered_rows)
```

Save the file

```
# Save the DataFrame to a CSV file
mod7_intersect_df.to_csv('N1-N2-V4.csv', index=False)
```