Geospatial Data Preparation for İzmir Green Space and Public Transportation Integration Analysis

1. Introduction

This study aims to analyze the **availability and distribution of green spaces in İzmir**, focusing on their proximity to different neighborhoods. The dataset for İzmir's green spaces was obtained in **CSV format from the İzmir Open Data Portal**, containing only text-based addresses instead of geographic coordinates. To integrate this data into a GIS environment, we must first **convert these addresses into latitude and longitude coordinates.** This report outlines the data preparation process, including:

1. **Geocoding** green space addresses into latitude and longitude,
2. **Converting** the dataset into a geospatial format (Shapefile, GeoJSON, GPKG),
3. **Ensuring** compatibility with GIS tools such as QGIS and Python's GeoPandas,
4. **Preparing the dataset** for further spatial analysis on **green space accessibility.**

2. Data Processing Steps

Step 1: Converting Addresses to Geographic Coordinates (Geocoding)

The first step is to convert the address-based location data into **latitude and longitude** values. This is done using the geopy library, which queries OpenStreetMap's **Nominatim API** to retrieve geographic coordinates for each address.

In [1]:
```
from geopy.geocoders import Nominatim
import pandas as pd
import time
```

In [2]:
```
# Initialize the geolocator
geolocator = Nominatim(user_agent="geo_app")
# Function to convert address to ccordinates
# Function to convert address to coordinates
def get_coordinates(address):
    try:
        location = geolocator.geocode(address)
        if location:
            return pd.Series([location.latitude, location.longitude])
        else:
            return pd.Series([None, None])
    except:
        return pd.Series([None, None])
```

In [3]:
```
# Load the CSV file containing green space data
df_north= pd.read_csv("kuzeyparklar.csv", sep=";")
df_south= pd.read_csv("guneyparklar.csv", sep=";")
```

In [4]:
```
df_north.head(2)
```

Out[4]:

	YESIL_ALAN_TURU	ILCE	MAHALLE	ADRES	PARK_ADI	YESIL_ALAN_MIKTARI	KONDISYON_TAKIMI	OYU
0	Park	Bayraklı	Adalet Mahallesi	Manas Bulvarı No: 78	Adalet Mahallesi Muhtarlığı Yanı Parkı	1640	1.0	
1	Park	Bayraklı	75. Yıl Mahallesi	1620/39 Sokak	Bayraklı 75.Yıl Parkı	2130	NaN	

In [5]:
```
# Apply geocoding function to each address
df_north[['latitude', 'longitude']] = df_north['ADRES'].apply(get_coordinates)
df_south[['latitude', 'longitude']] = df_south['ADRES'].apply(get_coordinates)
```

In [6]:
```
# Save the updated dataset
df_north.to_csv("north_green_spaces_with_coordinates.csv", index=False)
df_south.to_csv("south_green_spaces_with_coordinates.csv", index=False)
```

Explanation

- The script reads the green space dataset from the CSV file.
- It queries OpenStreetMap's API to obtain latitude and longitude for each address.
- The new coordinate values are stored in the dataset, which is saved as green_spaces_with_coordinates.csv for further processing.

Step 2: Converting CSV Data into a Spatial Dataset

Once we have latitude and longitude values, we need to convert the dataset into a geospatial format. This will allow us to perform spatial operations, such as identifying neighborhoods with limited green space availability.

In [7]:
```
import geopandas as gpd
from shapely.geometry import Point
```

In [8]:
```
# Load the CSV file with coordinates
df_north = pd.read_csv("north_green_spaces_with_coordinates.csv")
df_south = pd.read_csv("south_green_spaces_with_coordinates.csv")
```

In [9]:
```
# Create a geometry column with Point objects
geometry_north = [Point(xy) for xy in zip(df_north.longitude, df_north.latitude)]
gdf_north_green_spaces = gpd.GeoDataFrame(df_north, geometry=geometry_north, crs="EPSG:4326")

geometry_south = [Point(xy) for xy in zip(df_south.longitude, df_south.latitude)]
gdf_south_green_spaces = gpd.GeoDataFrame(df_south, geometry=geometry_south, crs="EPSG:4326")
```

In [10]:
```
# Save as a Shapefile
gdf_north_green_spaces.to_file("north_green_spaces.shp")
gdf_south_green_spaces.to_file("south_green_spaces.shp")
```

```
C:\Users\yalge\AppData\Local\Temp\ipykernel_20804\3964901530.py:2: UserWarning: Column names
longer than 10 characters will be truncated when saved to ESRI Shapefile.
  gdf_north_green_spaces.to_file("north_green_spaces.shp")
C:\Users\yalge\AppData\Local\Temp\ipykernel_20804\3964901530.py:3: UserWarning: Column names
longer than 10 characters will be truncated when saved to ESRI Shapefile.
  gdf_south_green_spaces.to_file("south_green_spaces.shp")
```

Explanation

- The script reads the updated CSV file and converts latitude/longitude pairs into geometric point objects using shapely.
- A GeoDataFrame is created using geopandas, with the EPSG:4326 coordinate system (WGS 84).
- The dataset is then saved as a Shapefile, making it compatible with GIS software like QGIS

Step 3: Handling Shapefile Column Name Limitations

The ESRI Shapefile format restricts column names to 10 characters, which may result in truncation and loss of clarity. To prevent this, we rename long column names.

In [11]:
```
# Rename long column names to fit the 10-character Shapefile limit
gdf_north_green_spaces.rename(columns={
    "latitude": "lat",
    "longitude": "lon",
    "green_space_name": "gs_name",
    "neighborhood_name": "nbhd_name"
}, inplace=True)
```

In [12]:
```
gdf_south_green_spaces.rename(columns={
    "latitude": "lat",
    "longitude": "lon",
    "green_space_name": "gs_name",
    "neighborhood_name": "nbhd_name"
}, inplace=True)
```

In [13]:
```
# Save the adjusted dataset as a Shapefile
gdf_north_green_spaces.to_file("north_green_spaces_fixed.shp")
gdf_south_green_spaces.to_file("south_green_spaces_fixed.shp")
```

```
C:\Users\yalge\AppData\Local\Temp\ipykernel_20804\1375902828.py:2: UserWarning: Column names
longer than 10 characters will be truncated when saved to ESRI Shapefile.
  gdf_north_green_spaces.to_file("north_green_spaces_fixed.shp")
C:\Users\yalge\AppData\Local\Temp\ipykernel_20804\1375902828.py:3: UserWarning: Column names
longer than 10 characters will be truncated when saved to ESRI Shapefile.
  gdf_south_green_spaces.to_file("south_green_spaces_fixed.shp")
```

Explanation

- Long column names (e.g., green_space_name, neighborhood_name) are shortened to comply with Shapefile constraints.
- The dataset is saved again as a Shapefile with the modified column names, ensuring compatibility with GIS applications.

Step 4: Saving in Alternative Geospatial Formats

Since Shapefile has limitations, we also save the dataset in **GeoJSON** and **GPKG (Geopackage)** formats, which allow longer column names and support more advanced geospatial queries.

In [14]:
```
# Save as GeoJSON (widely used for web applications)
gdf_north_green_spaces.to_file("north_green_spaces.geojson", driver="GeoJSON")
gdf_south_green_spaces.to_file("south_green_spaces.geojson", driver="GeoJSON")
```

In [15]:
```
# Save as GPKG (supports larger datasets and long column names)
gdf_north_green_spaces.to_file("north_green_spaces.gpkg", driver="GPKG")
gdf_south_green_spaces.to_file("south_green_spaces.gpkg", driver="GPKG")
```

Explanation

- GeoJSON is useful for web mapping applications and online GIS tools.
- GPKG (Geopackage) is a modern alternative to Shapefile, supporting large datasets and long attribute names.

3. Conclusion and Next Steps

With these processing steps completed, we now have a fully geospatial dataset of green spaces in İzmir. This dataset will be used for spatial accessibility analysis, helping to evaluate:

- Green space availability per neighborhood
- Identification of underserved areas
- Proximity analysis for different population groups

This prepared dataset will serve as the foundation for evidence-based urban planning to ensure equitable access to green spaces in İzmir.

In [ ]: