Supplementary Materials: Generated Scripts for Key Tasks

Below we provide the full Python code generated by the system using our system for GIS analysis in Pesaro. These scripts correspond to Tasks 3–6.

Task 3: Building Height Map

```
[SUCCESS] Execution successful!
Generated and executed code:
import pandas as pd
import geopandas as gpd
from pathlib import Path
import matplotlib.pyplot as plt

def direct_solution():
    # Define output directory and create it if not exists
    output_dir = r'C:\Users\Utente\GeoGPT\Geo_DeepSeek\BH'
    Path(output_dir).mkdir(parents=True, exist_ok=True)

    # Read building data from CSV
    csv_path = 'Dataset/CSV_GIS_Pesaro/Buildings2005Pesaro.csv'
    buildings_csv = pd.read_csv(csv_path)

    # Read shapefile
    shp_path = 'Dataset/Dati_Pesaro/buildings.shp'
    buildings_gdf = gpd.read_file(shp_path)

    # Drop rows with missing values in required columns
    buildings_csv.dropna(subset=['id_build', 'height'], inplace=True)
    buildings_gdf.dropna(subset=['id_edifici', 'altezza'], inplace=True)

    # Rename columns for joining
    buildings_csv = buildings_csv.rename(columns={'id_build': 'id_edifici'})

    # Join CSV data with shapefile data
    merged_data = pd.merge(buildings_gdf, buildings_csv, on='id_edifici', suffixes=('_shp', '_csv'))

    # Convert back to GeoDataFrame
    gdf = merged_data.copy()
    gdf['geometry'] = merged_data.geometry

    # Set consistent CRS if necessary (assuming shapefile has correct projection)
    target_crs = gdf.crs # Use the existing CRS from the shapefile
    gdf.to_crs(target_crs, inplace=True)  # This is redundant if already in target_crs

    # Prepare for plotting - create height categories
    height_values = gdf['height']

    # Create figure and axis
    fig, ax = plt.subplots(figsize=(15, 10))

    # Plot the GeoDataFrame with color coding
    vmax = max(gdf['height'])
    vmin = min(gdf['height'])

    plot = gdf.plot(column='height',
                    cmap='Reds',
                    ax=ax,
                    edgecolor='black',
                    legend=True,
                    legend_kwds={'label': 'Height'},
                    vmax=vmax,
                    vmin=vmin)
```

```
    # Add title and labels
    plt.title('Building Heights in Pesaro')
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')

    # Save the map to output directory
    map_path = f'{output_dir}\\building_heights_map.png'
    plt.savefig(map_path, dpi=300)
    plt.close()

    # Save merged shapefile to output directory
    shp_output_path = f'{output_dir}\\merged_buildings.shp'
    gdf.to_file(shp_output_path)
```

Task 4: Historical Development Mapping

```
[SUCCESS] Execution successful!
Generated and executed code:
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

def generate_building_map():
    # Read CSV data
    csv_path = 'Dataset/CSV_GIS_Pesaro/Buildings2005Pesaro.csv'
    df_csv = pd.read_csv(csv_path)

    # Clean CSV data: Convert 'year_ctr' to numeric and remove invalid entries
    df_csv['annoctr'] = pd.to_numeric(df_csv['year_ctr'], errors='coerce')
    df_csv.dropna(subset=['annoctr'], inplace=True)
    df_csv = df_csv[df_csv['annoctr'].notna()]

    # Read shapefile data
    shp_path = 'Dataset/Dati_Pesaro/buildings.shp'
    gdf_shp = gpd.read_file(shp_path)

    # Clean shapefile data: Convert 'annoctr' to numeric and remove invalid entries
    gdf_shp['annoctr'] = pd.to_numeric(gdf_shp['annoctr'], errors='coerce')
    gdf_shp.dropna(subset=['annoctr'], inplace=True)
    gdf_shp = gdf_shp[gdf_shp['annoctr'].notna()]

    # Aggregate CSV data to one row per 'annoctr'
    df_csv_aggregated = df_csv.groupby('annoctr').first().reset_index()

    # Perform a left merge on 'annoctr' to preserve all buildings
    gdf_merged = gdf_shp.merge(df_csv_aggregated, on='annoctr', how='left')

    # Prepare data for plotting (example: create year categories)
    bins = [0, 1900, 1950, 2000, 2024]
    labels = ['<1900', '1900-1950', '1950-2000', '>2000']
    gdf_merged['year_category'] = pd.cut(gdf_merged['annoctr'], bins=bins, labels=labels)

    # Plot the data
    plt.figure(figsize=(10, 8))
    gdf_merged.plot(column='year_category', categorical=True, legend=True, markersize=5)
    plt.title('Building Construction Year Categories')
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')
    plt.show()
```

Task 5: Spatial Distribution by Neighbourhood

```
[SUCCESS] Execution successful!
Generated and executed code:
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point, Polygon
import os
import pandas as pd

def direct_solution():
    # Set output directory
    output_dir = r'C:\Users\Utente\Geo_DeepSeek\BN'
    os.makedirs(output_dir, exist_ok=True)

    # Read neighborhood shapefile
    neighborhoods_path = 'Dataset/Dati_Pesaro/neighborhoods.shp'
    neighborhoods_gdf = gpd.read_file(neighborhoods_path)

    # Read buildings shapefile
    buildings_path = 'Dataset/Dati_Pesaro/buildings.shp'
    buildings_gdf = gpd.read_file(buildings_path)

    # Check if projections match, if not convert to same projection
    if neighborhoods_gdf.crs != buildings_gdf.crs:
        buildings_gdf = buildings_gdf.to_crs(neighborhoods_gdf.crs)

    # Spatial join: assign buildings to neighborhoods based on 'within' predicate
    joined_buildings = gpd.sjoin(
        buildings_gdf,
        neighborhoods_gdf[['denominazi', 'geometry']],
        how='inner',
        predicate='within'
    )

    # Clean up the results by dropping duplicates and NaN values
    joined_buildings.dropna(subset=['denominazi'], inplace=True)

    # Create a figure with two subplots
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 10))

    # Plot neighborhoods on first axis
    neighborhoods_gdf.plot(
        column='denominazi',
        categorical=True,
        legend=True,
        edgecolor='black',
        ax=ax1
    )
    ax1.set_title('Neighborhoods')
    ax1.set_xlabel('Longitude')
    ax1.set_ylabel('Latitude')

    # Plot buildings on second axis
    joined_buildings.plot(
        column='denominazi',
        categorical=True,
        markersize=10,
        color='red',
        alpha=0.7,
        edgecolor='black',
        ax=ax2
    )
    ax2.set_title('Buildings')
    ax2.set_xlabel('Longitude')
    ax2.set_ylabel('Latitude')

    plt.tight_layout()
```

```
    plt.show()

    # Save the joined data
    joined_buildings.to_file(os.path.join(output_dir, "joined_buildings.shp"))

# Execute the function
direct_solution()
Outputs saved successfully in: C:\Users\Utente\Geo_DeepSeek\BN
Files: ['joined_buildings.cpg', 'joined_buildings.prj', 'joined_buildings.shp', 'joined_buildings.dbf', 'joined_buildings.shx']
```

Task 6: Buffer Distance from City Centre

```
[SUCCESS] Execution successful!
Generated and executed code:
import geopandas as gpd
import matplotlib.pyplot as plt

# Define the output directory path
output_dir = "Dataset/Dati_Pesaro/output"

# Create output directory if it doesn't exist
import os
os.makedirs(output_dir, exist_ok=True)

# Load neighborhood and building shapefiles into GeoDataFrames
neighborhoods_path = "Dataset/Dati_Pesaro/neighborhoods.shp"
buildings_path = "Dataset/Dati_Pesaro/buildings.shp"

neighborhoods = gpd.read_file(neighborhoods_path)
buildings = gpd.read_file(buildings_path)

# Ensure both layers are in the same CRS; here we use EPSG:4326 as an example
target_crs = "EPSG:4326"
if neighborhoods.crs != target_crs:
    neighborhoods = neighborhoods.to_crs(target_crs)
if buildings.crs != target_crs:
    buildings = buildings.to_crs(target_crs)

# Filter to get 'Centro Storico' neighborhood and create a 1 km buffer
centro_storico = neighborhoods[neighborhoods['denominazi'] == 'Centro Storico']
buffer_gdf = centro_storico.copy()

# Reproject to projected CRS for metric correctness (UTM 33N for Pesaro), then buffer in metres
buffer_gdf = buffer_gdf.to_crs(epsg=32633)
buffer_gdf["geometry"] = buffer_gdf.geometry.buffer(1000)  # 1000 m = 1 km
# Reproject back to WGS84 for display/output
buffer_gdf = buffer_gdf.to_crs(epsg=4326)

# Spatial join to find buildings within the buffer
buildings_in_buffer = gpd.sjoin(buildings, buffer_gdf, how='inner')

# Plotting setup
plt.figure(figsize=(10, 8))
base_map = neighborhoods.plot(color='white', edgecolor='black')
centro_storico.plot(ax=base_map, color='green', label='Centro Storico')
buffer_gdf.plot(ax=base_map, color='blue', label=f'{buffer_distance} Buffer', alpha=0.5)
buildings_in_buffer.plot(ax=base_map, color='blue', markersize=20, label='Buildings in Buffer')

plt.title('Buildings within Buffered Area')
plt.legend()
plt.show()
```