I. Library import

In [1]:
```
#Import the librairies
import ipywidgets as widgets
from ipywidgets import interact, interactive,HBox, VBox
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import geopandas as gpd
import geodatasets
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import folium
import matplotlib
import mapclassify
from folium.plugins import MarkerCluster
import os
```

In [2]:
```
#Read the data from 2024 and 2023 and concat them
data2 = pd.read_csv("data.csv",low_memory=False)
data1 = pd.read_csv("data2023.csv",low_memory=False)
data = pd.concat([data1, data2], ignore_index=True)
```

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1584106 entries, 0 to 1584105
Data columns (total 18 columns):
 #   Column                     Non-Null Count    Dtype
---  ------                     --------------    -----
 0   date_mutation              1584106 non-null  object
 1   nature_mutation            1584106 non-null  object
 2   valeur_fonciere            1584106 non-null  float64
 3   code_commune               1584106 non-null  object
 4   code_departement           1584106 non-null  object
 5   id_parcelle                1584106 non-null  object
 6   nombre_lots                1584106 non-null  int64
 7   code_type_local            1584106 non-null  float64
 8   nombre_pieces_principales  1584106 non-null  float64
 9   surface_terrain            1584106 non-null  float64
 10  longitude                  1584106 non-null  float64
 11  latitude                   1584106 non-null  float64
 12  tranche_valeur             1584106 non-null  object
 13  code_insee                 1584106 non-null  int64
 14  region_name                1584106 non-null  object
 15  plage_surface              1584106 non-null  object
 16  prix_m2                    1584106 non-null  float64
 17  mois_annee                 1584106 non-null  object
dtypes: float64(7), int64(2), object(9)
memory usage: 217.5+ MB
```

II. Création des fonctions

In [4]:
```
def selection_filters_and_analyses(df):
    # Presentation of the filters
```

```
type_local_map = {
    1: 'House',
    2: 'Apartment',
    3: 'Dependency (Isolated)',
    4: 'Industrial and commercial premises or similar'
}
df['type_local'] = df['code_type_local'].map(type_local_map)

departments = df['code_departement'].unique()
types_properties = df['type_local'].unique()

# Widgets for the filters
department_select = widgets.SelectMultiple(
    options=departments,
    value=[],
    description='Departments',
    layout=widgets.Layout(width='60%')
)
department_select.style.description_width = '100px'

types_properties = df['type_local'].unique()
types_properties_select = widgets.SelectMultiple(
    options=types_properties,
    value=[],
    description='Types of property',
    layout=widgets.Layout(width='60%')
)
types_properties_select.style.description_width = '100px'

price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150k - 200k €",

price_range_select = widgets.SelectMultiple(
    options=price_range,
    value=[],
    description='Price range',
    layout=widgets.Layout(width='60%')
)
price_range_select.style.description_width = '100px'

slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200 m²', '200-25
slice_surface_select = widgets.SelectMultiple(
    options=slice_surface,
    value=[],
    description='Surface slice',
    layout=widgets.Layout(width='60%')
)
slice_surface_select.style.description_width = '100px'

# Widget to sekect the analyse to execute
disponible_analyses = {
    "Average price per m²": show_average_price_per_m2,
    "Average surface area" : show_average_surface_per_dept,
    "Price evolution" : evolution_average_price_per_m2_per_departement_month
    "Number of main parts" :display_number_of_room_department,
    "Type of property by department" : distribution_type_property_by_departm
    "Distribution of land values" : distribution_of_land_value_by_tranche,
    "Price per m² depending on the surface area" : price_m2_per_surface,
    "Explore the map" : explore_france

}
```

```
analyse_select = widgets.SelectMultiple(
    options=list(disponible_analyses.keys()),
    description='Analyse',
    layout=widgets.Layout(width='60%')
)

analyse_select.style.description_width = '100px'

# Button to apply the filters
bouton_apply = widgets.Button(description="Apply the filters", button_style=

# Function to take the filters and execute the analyse
def apply_filter(_):
    selected_analyses = list(analyse_select.value)

    # We execute each analyse selected
    for analyse in selected_analyses:
        func = disponible_analyses[analyse]
        func(df, department_select, types_properties_select,price_range_sele

# Link the button to the function
bouton_apply.on_click(apply_filter)

# Show widgets
display(department_select, types_properties_select,price_range_select,slice_
```

In [5]:

```
def show_average_price_per_m2(df,department_select,types_properties_select,price

    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)

    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150
    if not selected_slice_surface:
        selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200

    # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    df_filtered_national = df[
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    # Calculate the average price per m² by department
    average_price_par_dept = df_filtered.groupby('code_departement')['prix_m2'].

    # Calculate the average price of the selected departments
```

```
    average_price_select = average_price_par_dept.mean()

    # Calculate the national average price per m²
    national_average_price = df_filtered_national[df_filtered_national['type_loc

    # we create the graph
    fig, ax = plt.subplots(figsize=(20, 6))

    # We draw a bar to show the average price for each department
    ax.bar(average_price_par_dept.index, average_price_par_dept)

    # Added a bar for the average price selected
    ax.bar('Selected Average', average_price_select, alpha=0.7)

    # Added a bar for the national average price
    ax.bar('National Average', national_average_price, alpha=0.7)

    # And this is the parameters of the graphics
    ax.set_ylabel('Average price per m² (€)')
    ax.set_xlabel('Departments')
    ax.set_title("Average price per m² per department, selected and glogal avera
    ax.grid(True, linestyle='--', alpha=0.6)
    plt.xticks(rotation=90)
    plt.tight_layout()  # To avoid labels overlapping
    plt.show()
```

In [6]:

```
def show_average_surface_per_dept(df,department_select,types_properties_select,p

    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)

    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150
    if not selected_slice_surface:
        selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200

    # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    df_filtered_national = df[
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy


    # We calculate the average surface area for each selected department
    average_area_par_dept = df_filtered.groupby('code_departement')['surface_ter

    # Calculate the average surface area of the selected departments
```

```
    surface_moyenne_selectionnee = average_area_par_dept.mean()

    # Calculate the national average surface area (average of all selected depar
    national_average_area = df_filtered_national[df_filtered_national['type_loca

    fig, ax = plt.subplots(figsize=(20, 6))

    # We plot the average area for each selected department
    ax.bar(average_area_par_dept.index, average_area_par_dept, label='Average Su

    # Add a bar for the average area of the selected departments
    ax.bar('Selected Average', surface_moyenne_selectionnee, label='Selected Ave

    # Add a bar for the national average area
    ax.bar('National average', national_average_area, label='National average',

    # parameters of the graphics
    ax.set_ylabel('Average surface area in m²')
    ax.set_xlabel('Departments')
    ax.set_title("Average surface area in m² by department, selection and global
    ax.grid(True, linestyle='--', alpha=0.6)
    ax.legend(title="Legend", bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```

In [7]:

```
def evolution_average_price_per_m2_per_departement_monthly(df,department_select,
    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)

    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150
    if not selected_slice_surface:
        selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200

   # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    df_filtered_national = df[
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    # we verify that the date column is in the correct format
    df_filtered['date_mutation'] = pd.to_datetime(df_filtered['date_mutation'])

    # we take the average price per department for each month
    average_price_par_dept_mois = df_filtered.groupby(['code_departement', 'mois
```

```
    plt.figure(figsize=(15, 8))

    # Add the curve for the global average
    df_dept_global = df_filtered.groupby('mois_annee')['prix_m2'].mean().reset_i
    plt.plot(df_dept_global['mois_annee'].astype(str), df_dept_global['prix_m2']

    # Add the curve for the national average (based on the entire data)
    df_national_global = df_filtered_national.groupby('mois_annee')['prix_m2'].m
    plt.plot(df_national_global['mois_annee'].astype(str), df_national_global['p

    # Add the curves for each selected department
    for departement in selected_departments:
        df_dept = average_price_par_dept_mois[average_price_par_dept_mois['code_
        plt.plot(df_dept['mois_annee'].astype(str), df_dept['prix_m2'], marker='

    plt.title("Evolution of the average price per m² by department (monthly)")
    plt.xlabel("Month and year")
    plt.ylabel("Average price per m² (€)")
    plt.legend(title="Types of property", bbox_to_anchor=(1.05, 1), loc='upper l
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

In [8]:

```
def display_number_of_room_department(df,department_select,types_properties_sele
    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)

    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150
    if not selected_slice_surface:
        selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200

     # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    df_filtered_national = df[
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    # We take the number of main room per department
    number_piece_per_dept = df_filtered.groupby('code_departement')['nombre_piec

    # Ajouter la moyenne générale des départements sélectionnés
    average_selection = df_filtered['nombre_pieces_principales'].mean()
    national_average = df_filtered_national['nombre_pieces_principales'].mean()
```

```
    # Ajouter une barre pour la moyenne générale
    departments = list(number_piece_per_dept['code_departement'])
    number_room = list(number_piece_per_dept['nombre_pieces_principales'])

    plt.figure(figsize=(15, 8))
    plt.bar(departments, number_room, color='skyblue')
    #We plot line for average and national average of main rooms
    plt.axhline(average_selection, color='red', linestyle='dashed', linewidth=2,
    plt.axhline(national_average, color='green', linestyle='dashed', linewidth=2

    plt.title("Average number of rooms per department and per type of property s
    plt.xlabel("Department")
    plt.ylabel("Average number of rooms")
    plt.xticks(rotation=90)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
```

In [9]:

```
def distribution_type_property_by_department(df,department_select,types_properti
    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)

   # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = list(df['tranche_valeur'].unique())
    if not selected_slice_surface:
        selected_slice_surface = list(df['plage_surface'].unique())

     # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                     (df['type_local'].isin(selected_types_properties))&
                     (df['tranche_valeur'].isin(selected_price_range))&
                     (df['plage_surface'].isin(selected_slice_surface))].copy

    # Calculate the distribution of property types by department
    distribution = df_filtered.groupby(['code_departement', 'type_local']).size(

    # Normalize to obtain proportions (in %)
    distribution_norm = distribution.div(distribution.sum(axis=1), axis=0) * 100

    # creating a Stacked Bar Chart
    distribution_norm.plot(kind='bar', figsize=(15, 8), stacked=True, colormap='

    plt.title("Distribution of transactions by type of property and by departmen
    plt.xlabel("Department")
    plt.ylabel("Proportion of transactions (%)")
    plt.legend(title="Types of properties", bbox_to_anchor=(1.05, 1), loc='upper
    plt.grid(axis='y', linestyle='--', alpha=0.6)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

In [10]:
```
def distribution_of_land_value_by_tranche(df, department_select, types_propertie
    # Liste ordonnée des plages de prix
    price_order = [
        "< 50k €", "50k - 100k €", "100k - 150k €", "150k - 200k €", "200k - 250
        "250k - 300k €", "300k - 350k €", "350k - 400k €", "400k - 450k €", "450
        "500k - 550k €", "550k - 600k €", "600k - 650k €", "650k - 700k €", "700
        "750k - 800k €", "800k - 850k €", "850k - 900k €", "900k - 950k €", "950
        "1M - 1.5M €", "1.5M - 2M €", "2M - 2.5M €", "2.5M - 3M €", "3M - 4M €",
    ]

    # Transformer la colonne 'tranche_valeur' en catégorie ordonnée
    df['tranche_valeur'] = pd.Categorical(df['tranche_valeur'], categories=price

    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)

    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = price_order
    if not selected_slice_surface:
        selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200
                                  '250-300 m²', '300-350 m²', '350-400 m²', '400
                                  '500-600 m²', '600-700 m²', '700-800 m²', '800

    # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                     (df['type_local'].isin(selected_types_properties)) &
                     (df['tranche_valeur'].isin(selected_price_range)) &
                     (df['plage_surface'].isin(selected_slice_surface))].copy()

    # Count the number of transactions by tranche and by department
    distribution = df_filtered.groupby(['code_departement', 'tranche_valeur']).s
    # Normalize to obtain proportions (in %)
    distribution_norm = distribution.div(distribution.sum(axis=1), axis=0) * 100
    # Create a graphic
    distribution_norm.T.plot(kind='bar', figsize=(15, 8), stacked=True, colormap
    plt.title("Distribution of transactions by land value bracket and by departm
    plt.xlabel("Land value range (€)")
    plt.ylabel("Proportion of transactions (%)")
    plt.legend(title="Department", bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.grid(axis='y', linestyle='--', alpha=0.6)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

In [11]:
```
def price_m2_per_surface(df, department_select, types_properties_select,price_ra
    # We take the department,price range, slice surface and types of properties
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)
```

```
    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(df['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(df['type_local'].unique())
    if not selected_price_range:
        selected_price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150
    if not selected_slice_surface:
        selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200

     # Filter data based with our filter
    df_filtered = df[(df['code_departement'].isin(selected_departments)) &
                        (df['type_local'].isin(selected_types_properties))&
                        (df['tranche_valeur'].isin(selected_price_range))&
                        (df['plage_surface'].isin(selected_slice_surface))].copy

    df_filtered_national = df[
                        (df['type_local'].isin(selected_types_properties))&
                        (df['tranche_valeur'].isin(selected_price_range))&
                        (df['plage_surface'].isin(selected_slice_surface))].copy

    # Calculate the average price by surface area and department
    average_price_per_surface = df_filtered.groupby(['plage_surface', 'code_depa

     # Calculate average by surface range for selection
    average_selection = df_filtered.groupby('plage_surface')['prix_m2'].mean().r

    # Calculate the national average by surface area
    national_average = df_filtered_national.groupby('plage_surface')['prix_m2'].

    # Creating the graph
    plt.figure(figsize=(12, 8))
    #For each department selected, we take the average price per suface and then
    for department in selected_departments:
        df_department = average_price_per_surface[average_price_per_surface['cod
        if not df_department.empty:
            plt.plot(
                df_department['plage_surface'],
                df_department['prix_m2'],
                marker='o',
                label=f'Department {department}'
            )
    plt.plot(
        average_selection['plage_surface'],
        average_selection['prix_m2'],
        marker='o',
        linestyle='--',
        label='Selected average',
        color='black'
    )

    # Then we add the National average line
    plt.plot(
        national_average['plage_surface'],
        national_average['prix_m2'],
        marker='o',
        linestyle=':',
        label='National average',
        color='red'
    )
```

```
    # graph configurations
    plt.title("Average price per m² depending on surface area ranges")
    plt.xlabel("Surface range (m²)")
    plt.ylabel("Average price per m² (€)")
    plt.legend(title="Departement", bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

In [12]:
```
def add_region_name_column(df):
    """
    Add a column 'region_name' to the dataframe with the name of the region corr

    :param df: DataFrame with the column 'code_insee' containing the INSEE codes

    :return: DataFrame with the added column 'region_name'
    """
    # Dictionary of INSEE codes for regions and their names
    regions_names = {
        "01": "Guadeloupe", "02": "Martinique", "03": "Guyane", "04": "La Réunio
        "24": "Centre-Val de Loire", "27": "Bourgogne-Franche-Comté", "28": "Nor
        "44": "Grand Est", "52": "Pays de la Loire", "53": "Bretagne", "75": "No
        "84": "Auvergne-Rhône-Alpes", "93": "Provence-Alpes-Côte d'Azur", "94":
    }
    df['region_name'] = df['code_insee'].map(regions_names)
    return df
```

In [13]:
```
def explore_france(data, department_select=None, types_properties_select=None, p
    """
    Displays a map of France with the average price per m² of land by region, an

    Parameters:
    - data : pd.DataFrame : The DataFrame containing the data of real estate tra
    - department_select : ipywidgets.SelectMultiple : The widget for selecting d
    - types_properties_select : ipywidgets.SelectMultiple : The widget for selec
    - price_range_select : ipywidgets.SelectMultiple : The widget for selecting
    - slice_surface_select : ipywidgets.SelectMultiple : The widget for selectin

    Returns:
    - Map of France with the average price per m² of land by region, and the loc

    """
    data_general = data.copy()

    #We take the department,price range, slice surface and types of properties s
    selected_departments = list(department_select.value)
    selected_types_properties = list(types_properties_select.value)
    selected_price_range = list(price_range_select.value)
    selected_slice_surface = list(slice_surface_select.value)
    # If nothing is selected we take all the values for each point
    if not selected_departments:
        selected_departments = list(data['code_departement'].unique())
    if not selected_types_properties:
        selected_types_properties = list(data['type_local'].unique())
    if not selected_price_range:
        selected_price_range = ["< 50k €", "50k - 100k €", "100k - 150k €", "150
    if not selected_slice_surface:
```

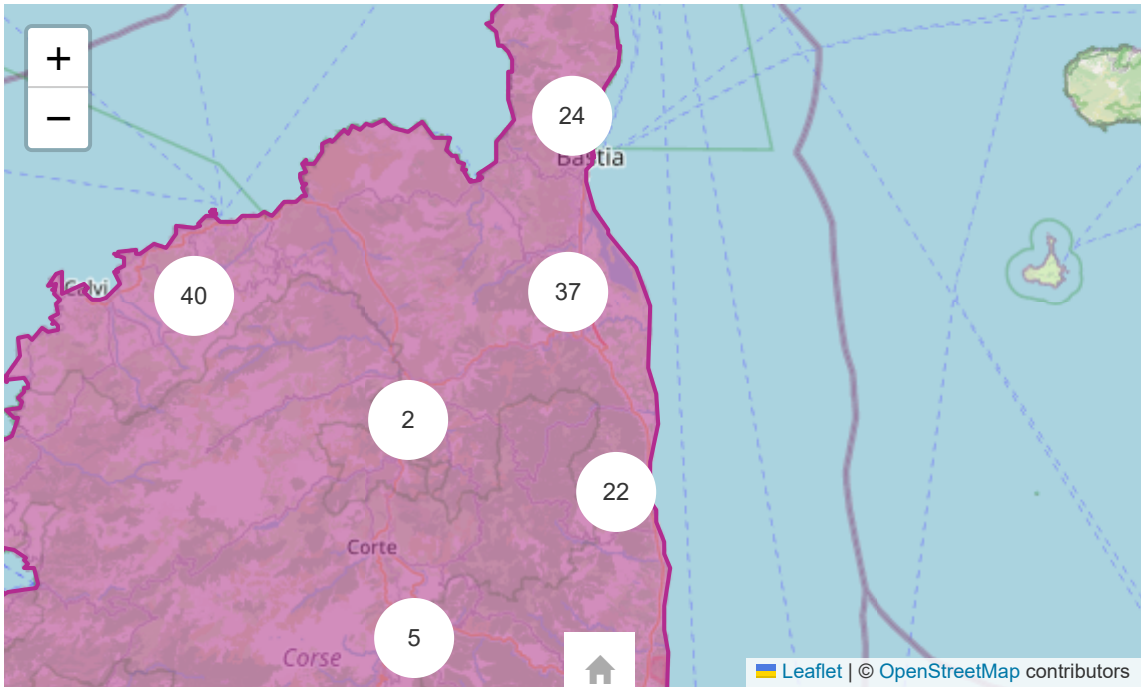
```
    selected_slice_surface = ['0-50 m²', '50-100 m²', '100-150 m²', '150-200

# Filter data based with our filter
data = data[
    (data['code_departement'].isin(selected_departments)) &
    (data['type_local'].isin(selected_types_properties)) &
    (data['tranche_valeur'].isin(selected_price_range)) &
    (data['plage_surface'].isin(selected_slice_surface))
].copy()

# Load shapefile
shp_file_path = "regions.shx"
map_france = gpd.read_file(shp_file_path)

# Define the CRS
if map_france.crs is None:
    map_france = map_france.set_crs(epsg=4326, allow_override=True)

#We verify that the column 'code_insee' is in the correct format and correct
data_general["code_insee"] = data_general["code_insee"].astype(str).str.zfil

#Average price per m² by region
ratio_average = data_general.groupby('code_insee')['prix_m2'].mean()

# Add the average land value by region stylized in euros
map_france['Average_land_value'] = map_france['code_insee'].map(ratio_averag
map_france['Average_land_value'] = map_france['Average_land_value'].apply(la


# Normalize the values
ratio_average = (ratio_average - ratio_average.min()) / (ratio_average.max()

# Logarithmic transformation to accentuate differences
ratio_average = np.log1p(ratio_average * 10)

map_france = map_france.set_index('code_insee').join(ratio_average, how='lef
map_france = add_region_name_column(map_france)

sample_data = data.copy() if department_select and types_properties_select a

geometry = gpd.GeoSeries.from_xy(sample_data['longitude'], sample_data['lati
points_gdf = gpd.GeoDataFrame(sample_data, geometry=geometry)

# Creating the map
map = folium.Map(location=[46.603354, 1.888334], zoom_start=6)
map_france["prix_m2"].fillna(0, inplace=True)
# We add the regions to the map
folium.GeoJson(
    map_france,
    name='geojson',
    style_function=lambda feature: {
        'fillColor':  matplotlib.colors.rgb2hex(plt.cm.plasma(feature['prope
        'color': matplotlib.colors.rgb2hex(plt.cm.plasma(feature['properties
        'weight': 2,
        'fillOpacity': 0.5,
    },
    tooltip=folium.GeoJsonTooltip(fields=['region_name', 'Average_land_value

).add_to(map)
```

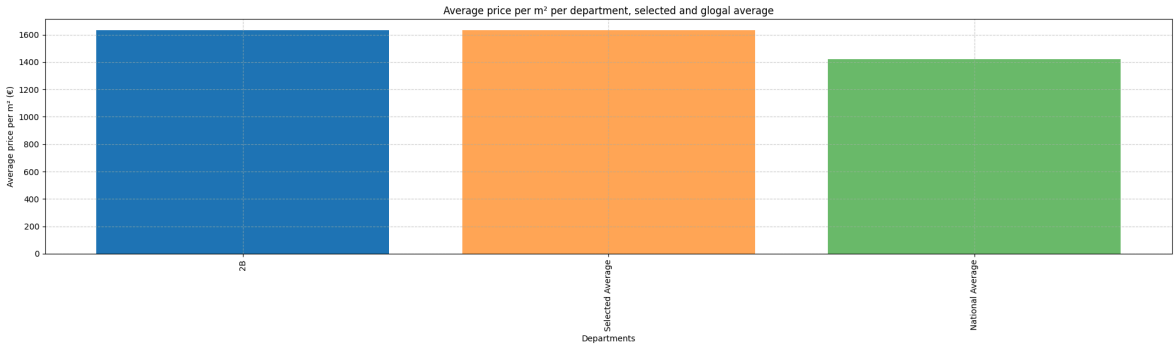
```
    # Add some points
    marker_cluster = MarkerCluster().add_to(map)

    for idx, row in points_gdf.iterrows():
        folium.Marker(
            location=[row['latitude'], row['longitude']],
            popup=f"{row.get('region_name', 'N/A')} \n Type : {row.get('type_loc
            icon=folium.Icon(color='red', icon='home')
        ).add_to(marker_cluster)

    display(map)
    return map
```

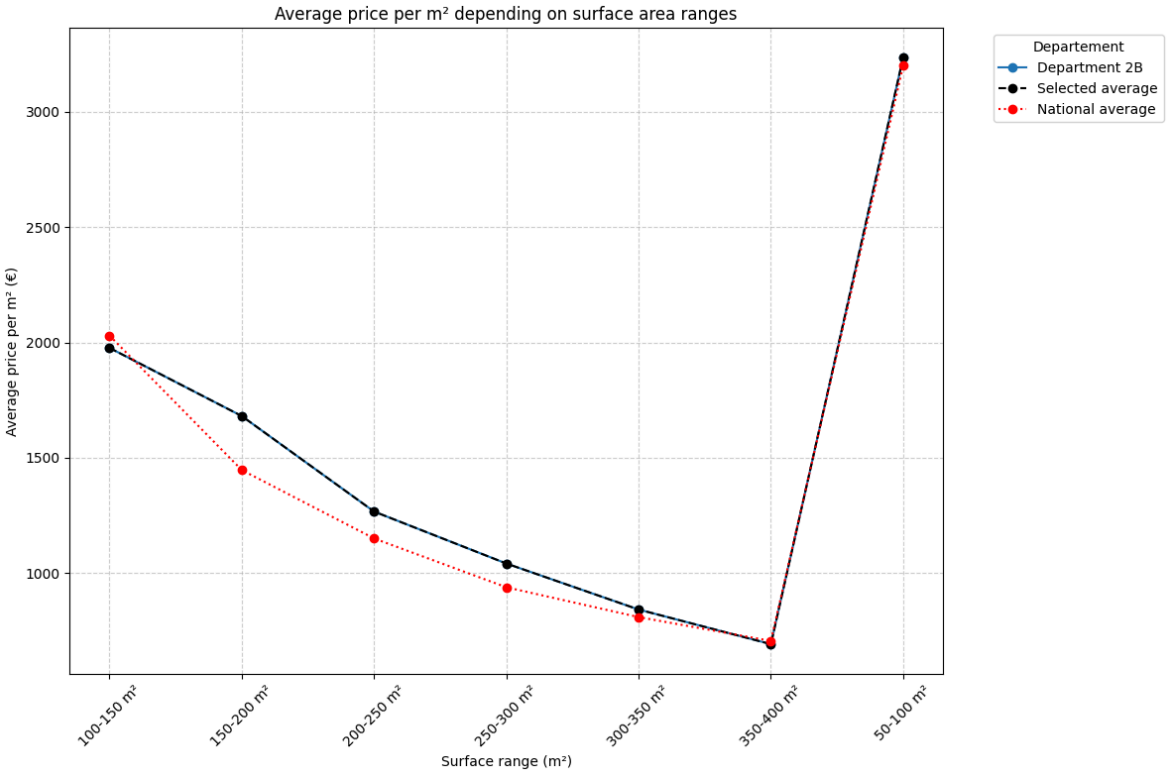
III. Exécution des fonctions

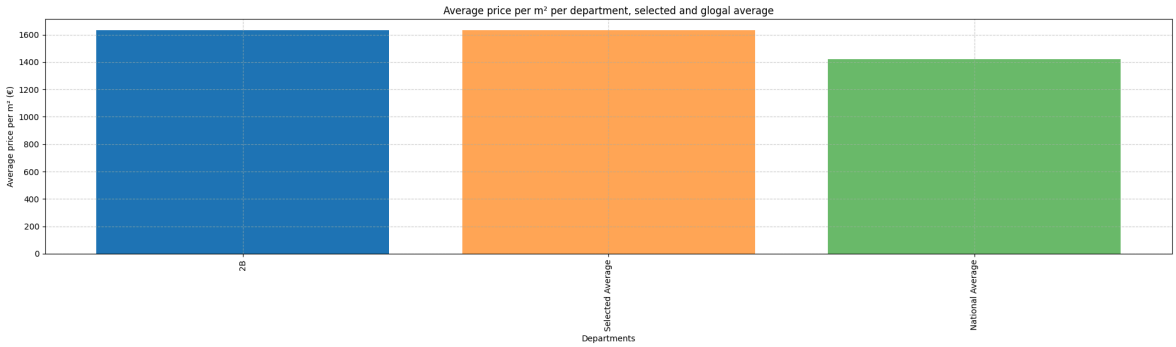
In [16]:
```
selection_filters_and_analyses(data)
```

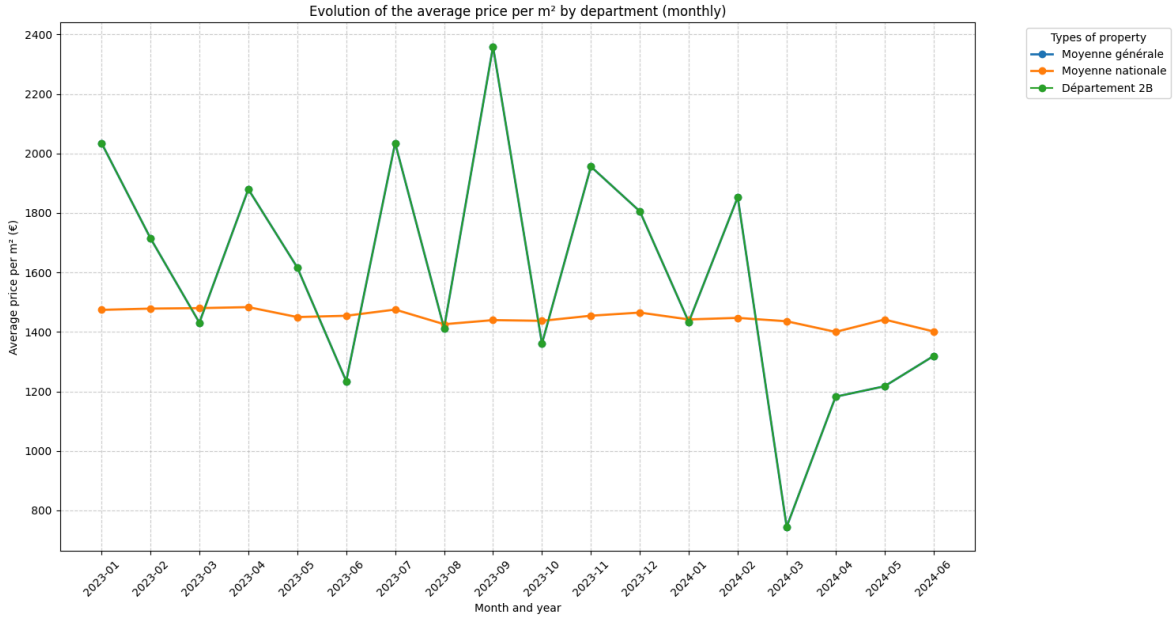
```
SelectMultiple(description='Departments', layout=Layout(width='60%'), options=('0
1', '02', '03', '04', '05', '…
SelectMultiple(description='Types of property', layout=Layout(width='60%'), optio
ns=('House', 'Dependency (Iso…
SelectMultiple(description='Price range', layout=Layout(width='60%'), options=('<
50k €', '50k - 100k €', '100…
SelectMultiple(description='Surface slice', layout=Layout(width='60%'), options=
('0-50 m²', '50-100 m²', '100-…
SelectMultiple(description='Analyse', layout=Layout(width='60%'), options=('Avera
ge price per m²', 'Average su…
Button(button_style='success', description='Apply the filters', style=ButtonStyle
())
C:\Users\Arthur\AppData\Local\Temp\ipykernel_25200\3141295893.py:76: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  map_france["prix_m2"].fillna(0, inplace=True)
```

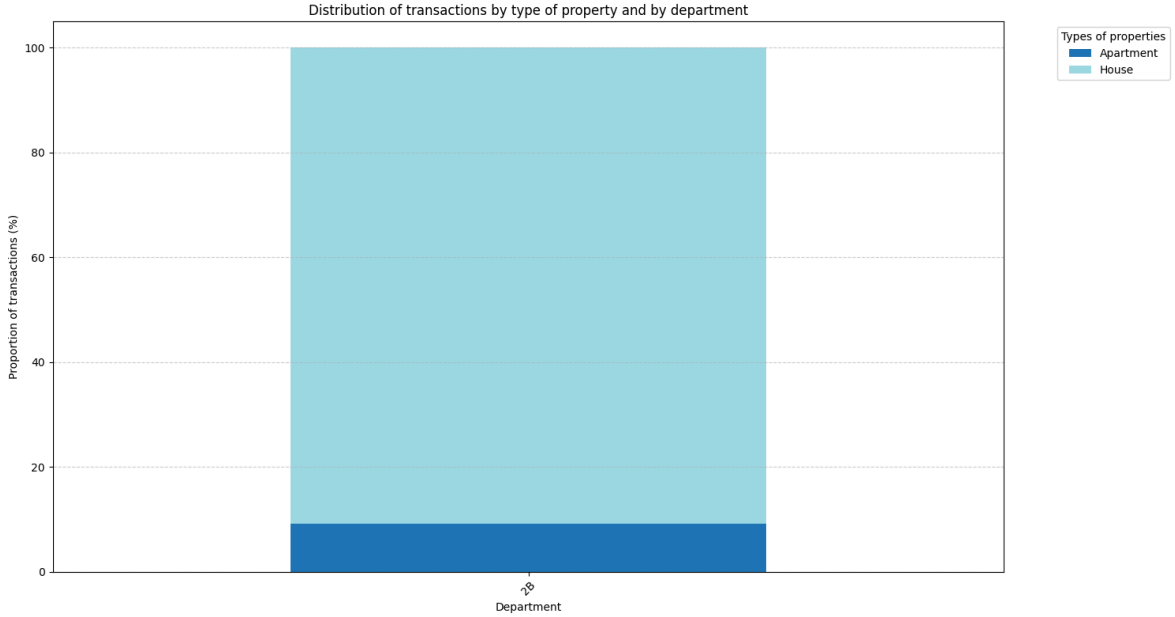

24
Bastia
40
37
Calvi
2
22
Corte
5
Corse
Leaflet | © OpenStreetMap contributors


Average price per m² per department, selected and glogal average
Average price per m² (€)
0
200
400
600
800
1000
1200
1400
1600
2B
Selected Average
National Average
Departments


Average price per m² depending on surface area ranges
Average price per m² (€)
1000
1500
2000
2500
3000
100-150 m²
150-200 m²
200-250 m²
250-300 m²
300-350 m²
350-400 m²
50-100 m²
Surface range (m²)
Departement
Department 2B
Selected average
National average







The investor is looking for a property in North Corsica, close to the sea (department code: 2B) to generate location rands, he has a maximum budget of €450,000 but he does not know what type of property he can find for his criteria: In the first graph (interactive map) we can see a distribution of properties with our criteria, there are 153, most of the properties are close to the sea because in Corsica most of the cities are on the coast. There are more properties on the east coast and cheaper.

In the second graph we can see the average price per m2 in North Corsica compared to the national price and North Corsica is much more expensive certainly because it is a

beautiful destination and a seaside region.

In the third graph we can see the price ratio m2 by size of real estate in North Corsica, it decreases sharply, it is divided by 2-3 between 100m2 and 400m2. The investor must therefore buy a large surface area but it could be useless for rental.

In the 5th graph we can see that the average price is lower than the national average since 2024, which may be the best time to invest and buy real estate.

In the 6th graph we can see the distribution of property types, in North Corsica there are only a few apartments and many houses, so a house could be the best choice because: There are many houses so a lot of choice and tenants will prefer the house for privacy, space and garden.

In conclusion and according to our analysis, the best places to invest in real estate for our investor are Bastia (East coast) or Calvi (West coast) for the choice of the house, Calvi should be more expensive to buy but the investor should earn more profits with the location. The real estate should be a house because it is the most popular and this house should have a garden if it is pleasant or will be easily because the price per m2 will be lower. The price of real estate depends on the city between Bastia and Calvi but it is between 200,000€ and 450,000€