Project 1

Introduction

The Great Recession in 2008 demonstrated how changes to the housing market have not only economic but also social consequences. John Bone and Karen O'Reilly (2010) suggest that the recent trend of viewing the purchase of property as an investment rather than a primary place to live contributes to growing housing unaffordability. This socio-economic issue specifically concerns younger people who do not come from a privileged background and do not have substantial savings to enter the housing market. According to Nissa Finney and Albert Sabater (2022), as housing in certain areas of England and Wales becomes less affordable, it causes a generational divide in housing opportunities. Age segregation becomes more prominent, presenting challenges to social mobility. Additionally, John Bone and Karen O'Reilly (2010) mention how a larger number of individuals owning a stable home results in more sustainable communities and benefits the overall wellbeing of many families. According to Nikodem Szumilo (2018), accessible homeownership would not necessarily change wages but might potentially have positive implications for increased economic activity.

McKee et al. (2016) discuss how different political narratives in England, Wales, Scotland and Northern Ireland lead to differences in housing polcies of local authorities. On November 22, 2017, the UK government announced a new policy, the First Time Buyers' (FTB) Relief, which exempted first time buyers from paying a Stamp Duty Land Tax (SDLT) on an acquired house with a value of less than 300 thousand pounds and removed property tax from house purchases valued less than 500 thousand. This policy targeted the lower end of the housing market and should have increased affordability for many young individuals who were trying to get on to the property ladder (Bolster, 2011). Housing is a necessity, as people require a place to live, and demand for most necessities is inelastic. When the government removes tax burden, demand increases, as individuals pay less. Economic theory predicts that any intervention that intends to decrease deadweight loss in the market with inelastic demand should not have a substantial effect on quantity but might largely increase the price, especially in a short run. The long term effect of the policy depends on the elasticity of supply. The supply of housing might increase because higher prices encourage individuals to sell their property and firms to build more housing units. However, the surge in supply happens to a lesser extent in densely populated areas, where the land for new construction projects is scarce. In the long run, housing prices might fall, depending on the increase in supply. As the UK government's policy targets the lower end of the market, the price and quantity changes should happen within property market for types and areas that are generally less expensive. Therefore, FTB Relief policy type requires clear evaluation.

This paper explores the impact of the First Time Buyers’ (FTB) Relief on housing prices of different property types in the Greater London area. Anna Bolster(2011) conducts analysis of the FTB Releif that was temporary introduced in 2011 but abolished later by HM of Reveue & Customs (p. 27). She concludes that the policy did not make housing more affordable and there was no signficant change in transactions, whereas tax relief was mostly substituted by the surge in prices. Shopov, Howell and Claridge(2023) similarly evaluated the FTB Relief implemented in 2017 but they concluded that in the £125,001 to £300,000 band the relief resulted in an 11% increase in transactions over and above the volume of FTB transactions that would have taken place in absence of the policy. According to Bryant (2012), FTB Relief does not offer the same level of benefit to individuals in different regions. For example, compared to other parts of the country, fewer people could use the tax break for purches of property in London because the avarage value of housing is much higher, even though the threshold for London was slightly bigger. Bryant (2012) also mentions how the effect of policy is minimized because many first-time buyers are not aware of legal costs and transactions fees that come with the purchase of a new home. This paper conducts further research into the effects of the 2017 policy on prices and investigates whether the housing market was affected disproportionally depending on the type of property and location.

This analysis uses the data provided by HM Land Registry, the non-ministerial department that monitors and documents every housing transaction in England and Wales and has an open database for all purchases since 1995 (HM Land Registry Open Data, 2023). This research paper attempts to evaluate how the FTB Relief influenced market prices and what difference it made for various types of housing. To monitor the policy effect, this exploration is limited to property transactions between March 2015 and March 2020, as the COVID-19 pandemic disrupted the economy overall. This research focuses on the Greater London area, the most populated English county with the most expensive housing in the region that accounts for almost 13% of all property transactions in England and Wales. The original data set is also merged with the data on green spaces in each borough, which was web scraped from the register of Historic England and its National Heritage List(Registered Parks & Gardens | Historic England, n.d.). The research also uses the data on modelled annual changes to population in each borough, as well as the set provided by Homes and Communities Agency and Local Authorities that present information about the amount of affordable housing supplied in each borough (Office for National Statistics, 2022). This paper evaluates the benefits of the policy, comparing its short-term and long-term effects on different housing types.

Data Cleaning

The data set used in this research includes 16 variables: Transaction Identifier (each purchase has a unique value), Price Paid, Date of Transfer, Postcode, Property Type, Old/New (New Build status), Tenure, Primary Address (PAON), Secondary Address (SAON), Street, Locality, City or Town, District, County, PPD_Category Type, and Record Status(HM Land Registry Open Data, 2023). All address variables indicate the location of the sold property. Property Type specifies if it is a flat (F), detached (D), semi-detached (S), or terraced (T) house, or other type of property (O). Tenure ('Duration') can be freehold (F) or leasehold (L), a trait common in the housing market of common law countries. PPD_Category indicates whether the purchase was a Standard Price Paid entry (A) or an Additional Price Paid entry (B).

The research question focuses on the period from March 2015 to March 2020, so transactions outside this timeframe are excluded, reducing data set from 28 276 227 to 5 180 168 values. The analysis is centered on residential properties, thus only transactions for detached, semi-detached, terraced houses, and flats are considered. The filtering process removes PAON, SAON, and Record Status columns, and data where the Tenure type is unknown, as they are irrelevant to the research question. It also excludes all purchases made outside of the Greater London Area, as it is the primary region of interest. These manipulations reduce the data set to 552 053 values and the number of observations stays the same after merging with data sets on parks and affordable housing supply.

The main focus of this research is to find the policy effect on the property prices, therefore making price the dependent variable. The policy requires a purchase to meet certain coditions for an individual to qualify for the tax relief, therefore Date of Transfer, Property Type, New Build status and District are independent variables that determine various sections of the housing market that could react differently to the government intervention. Date of Transfer can generate two other variables that will help analyze changes of housing prices over time. 'Before/After' variable indicates whether the property was purchased before or after the FTB Relief implementation on November 22, 2017. 'Interval_1' variable helps to group data for a time period of one year but setting the start point to March, as the latest data included is form March 2020. 'Interval_2' variable represents a time period with duration of 3 month and was created to help plot the change in prices in a shorter term. All entries with missing price values are removed. However, there are none.

New data sets add population, number of parks and affordable housing supply for each borough on the level of an individual house, showing which how many parks are located near the purchased property, how many people lived in the area during the time of purchase and how many affordable houses were constructed there at the time.

In [1]:
```
import numpy as np
import pandas as pd
import os
```

In [2]:
```
dataset_path = "/Users/user/Desktop/ECO225/ECO225Project/Data/202304.csv"
colnames = ['Transaction_unique_identifier', 'price', 'Date_of_Transfer',
            'postcode', 'Property_Type', 'Old/New',
            'Duration', 'PAON', 'SAON',
            'Street', 'Locality', 'Town/City',
            'District', 'County', 'PPDCategory_Type',
            'Record_Status - monthly_file_only']
```

In [3]:
```
# Read the dataset in chunks
chunk_size = 10000
df_chunks = pd.read_csv(dataset_path, header=None, names=colnames, chunksize=chunk_size)
```

In [4]:
```
#Convert "Date_of_Transfer" to year and month
df = pd.concat([chunk.assign(Year_of_Transfer=pd.to_datetime(chunk['Date_of_Transfer']).
                             Month_of_Transfer=pd.to_datetime(chunk['Date_of_Transfer'])
                             Day_of_Transfer=pd.to_datetime(chunk['Date_of_Transfer']).d
                for chunk in df_chunks])
```

In [5]:
```
df = df[((df['Year_of_Transfer'] == 2015) & (df['Month_of_Transfer'] >= 3)) |
        ((df['Year_of_Transfer'] > 2015) & (df['Year_of_Transfer'] < 2020)) |
        ((df['Year_of_Transfer'] == 2020) & (df['Month_of_Transfer'] < 3))]
```

In [6]:
```
# Remove unnecessary columns
df_copy = df.copy()
columns_to_remove = ['PAON', 'SAON', 'Record_Status - monthly_file_only']
df_copy = df_copy.drop(columns=columns_to_remove)
df_copy = df_copy[(df_copy['Duration'] != 'U')]
```

In [7]:
```
df_copy['Date_of_Transfer'] = pd.to_datetime(df_copy['Date_of_Transfer'])
reference_date = pd.to_datetime('2017-11-22') #date when policy was implemented
# Create the 'Before/After' variable
```

In [8]:
```
property_types_to_keep = ['D', 'S', 'T', 'F']
```

In [9]:
```
df_copy = df_copy[df_copy['Property_Type'].isin(property_types_to_keep) & (df_copy['Coun
```

In [10]:
```
groupby_vars = ['Property_Type', 'Before/After' 'Old/New', 'Duration', 'Day_of_Transfer'
```

In [11]:
```
intervals_1 = [
    ((2015, 3), (2016, 2)),
    ((2016, 3), (2017, 2)),
    ((2017, 3), (2018, 2)),
    ((2018, 3), (2019, 2)),
    ((2019, 3), (2020, 2))
]

intervals_2 = [
    ((2015, 3), (2015, 5)),
    ((2015, 6), (2015, 8)),
    ((2015, 9), (2015, 11)),
    ((2015, 12), (2016, 2)),
    ((2016, 3), (2016, 5)),
    ((2016, 6), (2016, 8)),
    ((2016, 9), (2016, 11)),
    ((2016, 12), (2017, 2)),
    ((2017, 3), (2017, 5)),
    ((2017, 6), (2017, 8)),
    ((2017, 9), (2017, 11)),
    ((2017, 12), (2018, 2)),
    ((2018, 3), (2018, 5)),
    ((2018, 6), (2018, 8)),
    ((2018, 9), (2018, 11)),
    ((2018, 12), (2019, 2)),
    ((2019, 3), (2019, 5)),
    ((2019, 6), (2019, 8)),
    ((2019, 9), (2019, 11)),
    ((2019, 12), (2020, 2)),
]
```

In [12]:
```
# Function to determine the interval name for a given year and month
def get_interval_name(start_year, end_year):
    return f"March {start_year} - February {end_year}"

def get_interval_name_2(start_month, start_year, end_month, end_year):
    return f"{start_month} {start_year} - {end_month} {end_year}"
```

In [13]:
```
# Function to determine the interval for a given year and month
def get_interval(year, month):
    for i, ((start_year, start_month), (end_year, end_month)) in enumerate(intervals_1):
        if (year > start_year or (year == start_year and month >= start_month)) and \
           (year < end_year or (year == end_year and month <= end_month)):
            return get_interval_name(start_year, end_year)

def get_interval_2(year, month):
    for i, ((start_year, start_month), (end_year, end_month)) in enumerate(intervals_2):
        if (year > start_year or (year == start_year and month >= start_month)) and \
           (year < end_year or (year == end_year and month <= end_month)):
            return get_interval_name_2(start_month, start_year, end_month, end_year)
```

In [14]:
```
# Apply the intervals used later for calculations
df_copy['Interval_1'] = df_copy.apply(lambda x: get_interval(x['Year_of_Transfer'], x['M
df_copy['Interval_2'] = df_copy.apply(lambda x: get_interval_2(x['Year_of_Transfer'], x[
```

In [15]:
```
intervals = ['3 2015 - 5 2015', '6 2015 - 8 2015', '9 2015 - 11 2015',
             '12 2015 - 2 2016','3 2016 - 5 2016', '6 2016 - 8 2016',
             '9 2016 - 11 2016', '12 2016 - 2 2017',
             '3 2017 - 5 2017', '6 2017 - 8 2017', '9 2017 - 11 2017',
             '12 2017 - 2 2018', '3 2018 - 5 2018',
             '6 2018 - 8 2018','9 2018 - 11 2018',
             '12 2018 - 2 2019', '3 2019 - 5 2019', '6 2019 - 8 2019',
             '9 2019 - 11 2019', '12 2019 - 2 2020']
```

In [16]:
```
df_copy['Before/After'] = df_copy['Date_of_Transfer'].apply(lambda x: 'Before' if x < re
```

Summary Statistics Table

In [17]:
```
grouped_df = df_copy.groupby(['County'])['price'].describe()
grouped_df.reset_index(inplace=True)

# Rename columns for consistency
grouped_df.rename(columns={'mean': 'Price_Mean', '50%': 'Price_Median', 'std': 'Price_St
grouped_df['Price_Std (Thousand £)'] /= 1000
grouped_df = pd.DataFrame(grouped_df)
grouped_df
```

Out[17]:

	County	Price_Count	Price_Mean	Price_Std (Thousand £)	min	25%	Price_Median	75%	max
0	GREATER LONDON	552053.0	598128.947179	817.025826	1.0	325000.0	440000.0	632050.0	160000000.0

In [18]:
```
grouped_df_type = df_copy.groupby(['County'])['Property_Type'].describe()
grouped_df_type['percentage'] = grouped_df_type['freq']/grouped_df_type['count'] * 100
grouped_df_type
```

Out[18]:

	count	unique	top	freq	percentage
County					
GREATER LONDON	552053	4	F	314878	57.03764

In [19]:
```
grouped_df_age = df_copy.groupby(['County'])['Old/New'].describe()
grouped_df_age['percentage'] = grouped_df_age['freq']/grouped_df_age['count'] * 100
grouped_df_age
```

Out[19]:

	count	unique	top	freq	percentage
County					
GREATER LONDON	552053	2	N	463107	83.888141

In [20]:
```
grouped_df_policy = df_copy.groupby(['County'])['Before/After'].describe()
grouped_df_policy['percentage'] = grouped_df_policy['freq']/grouped_df_policy['count'] *
grouped_df_policy
```

Out[20]:

	count	unique	top	freq	percentage
County					
GREATER LONDON	552053	2	Before	322228	58.369033

In [21]:
```
grouped_df_interval = df_copy.groupby(['County'])['Interval_1'].describe()
grouped_df_interval['percentage'] = grouped_df_interval['freq']/grouped_df_interval['cou
grouped_df_interval
```

Out[21]:

	count	unique	top	freq	percentage

County					
GREATER LONDON	552053	5	March 2015 - February 2016	127770	23.144517

In the Greater London area, property purchases exhibit a notable level of variability, as indicated by the substantial standard deviation of £843 thousand pounds. Despite this variation, the price of purchased properties stands at £598,128 on average, suggesting a central tendency within the market. However, the presence of extreme outliers, such as properties priced at £160 million pounds, underscores the existence of high-end segments within the market.

Furthermore, the upper quartile value of £642,500 pounds implies that a majority of purchased property fall below this threshold, reflecting a pronounced demand for more affordable housing options. This demand is further evidenced by the prevalence of flats as the most frequently purchased property type, which also reflects characteristics of the urban housing composition within a highly populated city.

Additionally, the higher frequency of purchases for old housing units compared to new ones suggests a preference for established properties.

Following policy implementation, there has been a marginal increase in the number of purchases, indicative of increased demanded quanitity. Notably, the peak in property purchases occurred between March 2015 and February 2016, predating the introduction of the UK government's First-Time Buyer Relief. These observations collectively paint a nuanced picture of the Greater London property market, characterized by varying price ranges, demand dynamics, and policy influences.

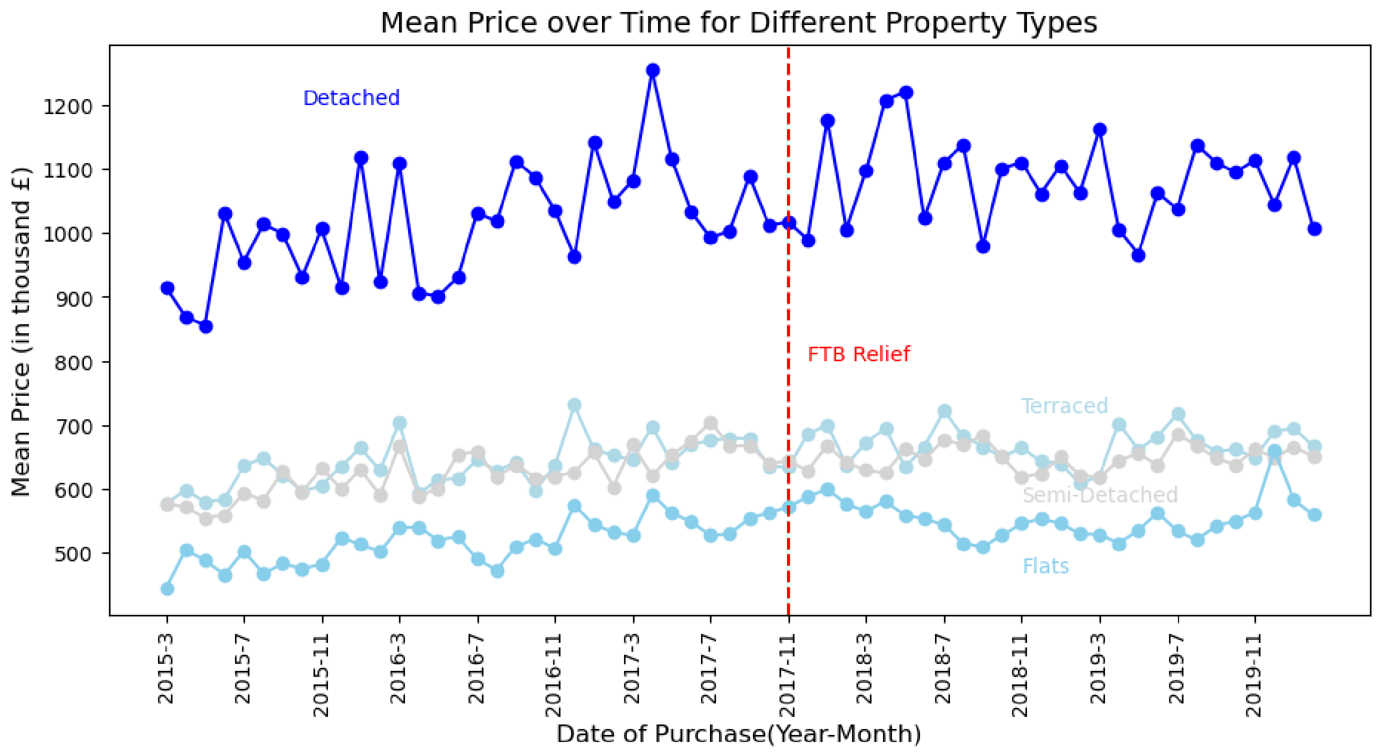
The following visualisations help to explore these tendencies further.

Plots, Histograms, Figures

In [22]:
```python
import matplotlib.pyplot as plt
```

In [23]:
```python
%matplotlib inline
```

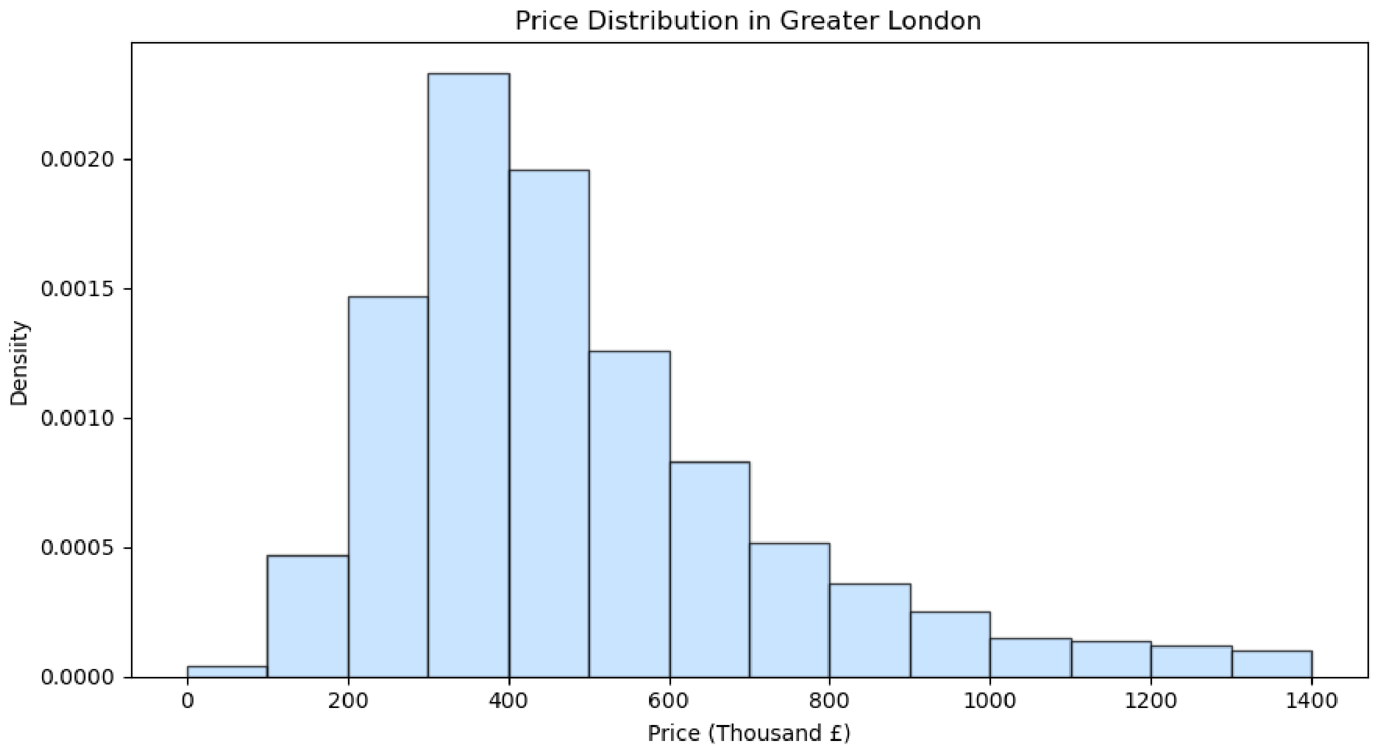
In [133…
```python
def plot_mean_price_over_time(df, property_types_to_keep, ax):
    color_map = {'D': 'blue', 'E': 'lightblue', 'F': 'skyblue', 'S': '#D3D3D3'}
    # Loop over each property type
    for prop_type in property_types_to_keep:
        df_filtered = df_copy[df_copy['Property_Type'] == prop_type].copy()
        # Convert 'Date_of_Transfer' to datetime
        df_filtered['Date_of_Transfer'] = pd.to_datetime(df_filtered['Date_of_Transfer']
        # Extract year and month from 'Date_of_Transfer'
        df_filtered['Year'] = df_filtered['Date_of_Transfer'].dt.year
        df_filtered['Month'] = df_filtered['Date_of_Transfer'].dt.month
        # Group by year and month, calculate the mean price for each group
        grouped_df = df_filtered.groupby(['Year', 'Month'])['price'].mean().reset_index(
        # Convert price to thousands of pounds
        grouped_df['price'] /= 1000
        line_color = color_map.get(prop_type, 'lightblue')
        ax.plot(grouped_df['Year'].astype(str) + '-' + grouped_df['Month'].astype(str),
        # Add labels on the lines
        if prop_type == 'D':
            ax.text('2015-10', 1200, 'Detached', fontsize=10, color=line_color)
        elif prop_type == 'T':
            ax.text('2018-11', 720, 'Terraced', fontsize=10, color=line_color)
        elif prop_type == 'S':
            ax.text('2018-11', 580, 'Semi-Detached', fontsize=10, color=line_color)
        elif prop_type == 'F':
            ax.text('2018-11', 470, 'Flats', fontsize=10, color=line_color)
    ax.axvline(x='2017-11', color='red', linestyle='--')
    ax.set_xlabel('Date of Purchase(Year-Month)', fontsize=12)  # Reduced fontsize for a
    ax.set_ylabel('Mean Price (in thousand £)', fontsize=12)  # Reduced fontsize for axi
    ax.set_title('Mean Price over Time for Different Property Types', fontsize=14)  # Re
    ax.tick_params(axis='x', rotation=90)  # Rotate x-axis labels
    ax.set_xticks(ax.get_xticks()[::4])
fig, ax = plt.subplots(figsize=(11, 5))
plt.text('2017-12', 800, 'FTB Relief', fontsize=10, color='red', rotation=0)
plot_mean_price_over_time(df, ['D', 'T', 'S', 'F'], ax)
plt.show()
```



This graph illustrates the average monthly price of properties sold across different housing types, providing valuable insights into the dynamics of the housing market. It is evident that detached houses typically command higher prices, accompanied by greater price volatility compared to flats, semi-detached, or terraced houses. The average value of flats, semi-detached, or terraced houses tends to be below the £600 thousand pounds mark, with flats exhibiting the lowest average price. The volatility observed in detached house prices can be attributed to seasonal fluctuations, particularly heightened demand during the spring months as families seek to purchase and relocate before the onset of summer (Ngai & Tenreyro, 2014). This surge in demand often triggers competitive bidding wars among prospective buyers, leading to temporary price escalations despite a consistent level of supply. The premium attached to detached houses reflects consumers' heightened willingness to pay for the privacy and autonomy offered by standalone properties, which do not share walls with neighbors.

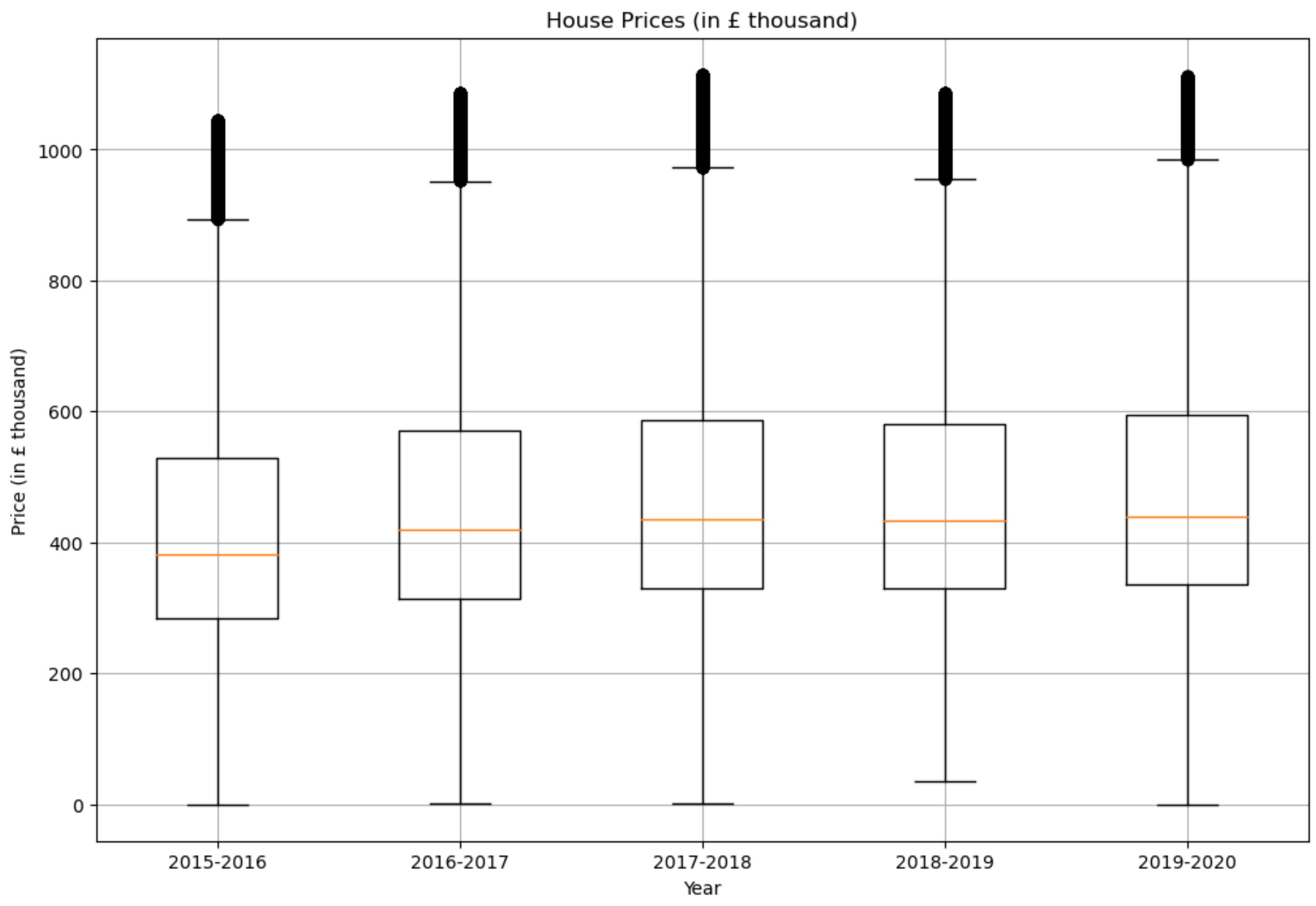
The red line on the graph indicates the implementation of the tax break. Notably, a significant shift in the graph occurs after November 2017, defying the typical seasonal decline in property prices associated with colder months marked by reduced sales activity. During this period, flats and terraced houses experienced a positive three-month change in average prices, coinciding with the introduction of the First-Time Buyer's Relief policy. This policy intervention likely have counteracted the usual market trend for these property types, while detached and semi-detached houses remained largely unaffected.

In [25]:
```
# Determine the 95th percentile of prices to identify outliers
price_95th_percentile = df_copy['price'].quantile(0.95)

# Filter the DataFrame to exclude outliers
df_filtered = df_copy[df_copy['price'] <= price_95th_percentile].copy()

# Convert prices to thousands for the filtered data
df_filtered.loc[:, 'price_thousands'] = df_filtered['price'] / 1000

# Determine the minimum and maximum prices in the filtered data set (in thousands)
min_price = df_filtered['price_thousands'].min()
max_price = df_filtered['price_thousands'].max()
bin_edges = list(range(int(min_price), int(max_price) + 100, 100))


plt.figure(figsize=(9, 5))
plt.hist(df_filtered['price_thousands'], bins=bin_edges, color='#b3d9ff', edgecolor='bla
plt.xlabel('Price (Thousand £)')
plt.ylabel('Densiity')
plt.title('Price Distribution in Greater London')
plt.tight_layout()
plt.show()
```



This graph vividly illustrates the distribution of property values in the Greater London area, highlighting a pronounced skew towards the left side of the price range. The majority of property units are valued below £600 thousand pounds, indicative of a robust demand for more affordable housing options within the market. Specifically, there is a notable concentration of houses and flats purchased in the price range of £300 to £400 thousand pounds, underscoring the prevalence of properties within this relatively lower price bracket. The data reflects the targeted approach of the First-Time Buyer's Relief policy, which aims to facilitate entry into the housing market for first-time buyers by focusing on properties priced below £500 thousand pounds. By implementing a cap on the qualifying property value, the UK government strategically ensures that it can continue to generate tax revenue from higher-priced transactions, thereby contributing to efforts aimed at reducing the national deficit. This policy intervention aligns with broader economic objectives of promoting homeownership among first-time buyers while maintaining fiscal sustainability.

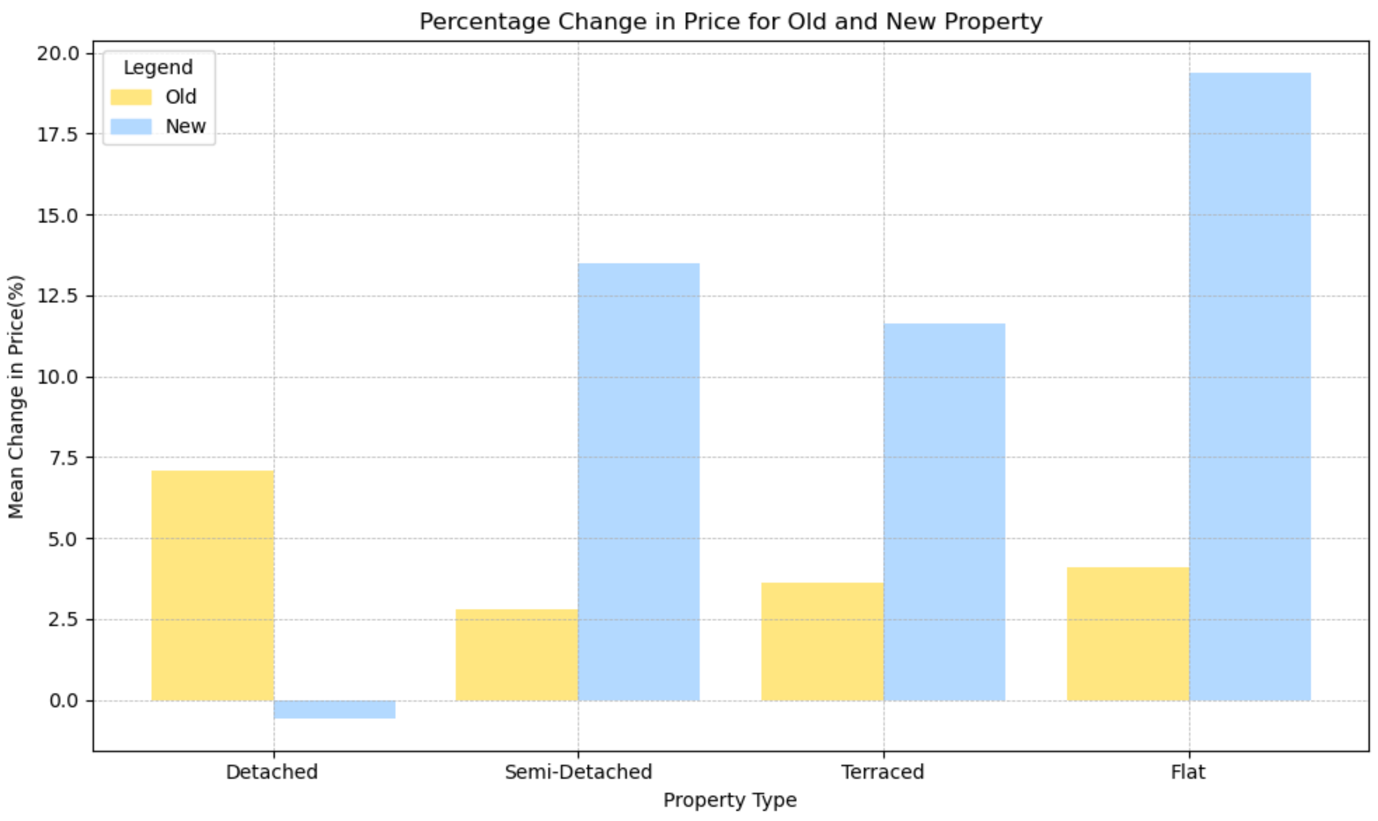
In [26]:
```python
interval_labels = ['March 2015 - February 2016',
                   'March 2016 - February 2017',
                   'March 2017 - February 2018',
                   'March 2018 - February 2019',
                   'March 2019 - February 2020']

# Initialize an empty list to store all data without outliers
all_data_no_outliers = []
years = []

for interval_label in interval_labels:
    # Filter the DataFrame for the specified interval
    df_interval = df_copy[df_copy['Interval_1'] == interval_label]
    years.append(interval_label.split()[1]+ '-'+ interval_label.split()[4])
    # Extract the price data for the interval and divide by 1000
    data = df_interval['price'] / 1000

    # Calculate the first and third quartiles
    q1, q3 = np.percentile(data, [25, 75])

    # Calculate the interquartile range (IQR)
    iqr = q3 - q1

    # Define the lower and upper bounds for outliers
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    # Remove outliers from the data
    data_no_outliers = data[(data >= lower_bound) & (data <= upper_bound)]

    # Add data without outliers to the list
    all_data_no_outliers.append(data_no_outliers)

# Create the box plot with all intervals on the same graph
plt.figure(figsize=(12, 8))
plt.boxplot(all_data_no_outliers, labels=years)


plt.title('House Prices (in £ thousand)')
plt.xlabel('Year')
plt.ylabel('Price (in £ thousand)')
plt.grid(True)
plt.show()
```



This box plot provides insights into the general trends observed in the median price of properties sold across various intervals, independent of other influencing variables. Notably, the data reveals a consistent upward trajectory in the typical price of properties sold from March 2015 to March 2017. However, the rate of increase moderated from March 2017 to March 2018, with minimal fluctuations in median values observed during subsequent intervals. The implementation of the First-Time Buyer's Relief Policy does not correspond to a significant deviation in prices or overall trends within the housing market. This observation underscores the need for further research to explore the specific impact of the policy, particularly in discerning which types of housing may have been most affected by its implementation.

In [27]:
```
# Calculate percentage change of mean price from one interval to the other
grouped_df = df_copy.groupby(['Property_Type', 'Before/After', 'Old/New'])
mean_price = grouped_df['price'].mean()
mean_price_unstacked = mean_price.unstack(level='Before/After')

# Calculate the percentage change
percentage_change = ((mean_price_unstacked['After'] - mean_price_unstacked['Before']) /
percentage_change = percentage_change.rename('Mean_Percent_Change')
percentage_change = percentage_change.reset_index()
```

In [28]:
```
fig, ax = plt.subplots(figsize=(10, 6))
type_order = percentage_change['Property_Type'].unique()
age = percentage_change['Old/New'].unique()
colors = {'Y': '#b3d9ff', 'N': '#ffe680'}
bar_width = 0.4

for i in range(len(type_order)):
    p_type = type_order[i]
    p_data = percentage_change[percentage_change['Property_Type'] == p_type]
    n_values = p_data[p_data['Old/New'] == 'N']['Mean_Percent_Change'].values
    y_values = p_data[p_data['Old/New'] == 'Y']['Mean_Percent_Change'].values
    ax.bar([i - bar_width / 2] * len(n_values), n_values, width=bar_width, color=colors[
    ax.bar([i + bar_width / 2] * len(y_values), y_values, width=bar_width, color=colors[

ax.set_xticks(range(len(type_order)))
ax.set_xticklabels(['Detached', 'Semi-Detached', 'Terraced', 'Flat'])
legend_handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in ['N', 'Y']]
ax.legend(legend_handles, ['Old', 'New'], title='Legend')
ax.set_ylabel('Mean Change in Price(%)')
ax.set_xlabel('Property Type')
ax.set_title('Percentage Change in Price for Old and New Property')
ax.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```



This graph presents a comparative analysis of the growth rates in average property prices for both old and new housing from March 2015 to March 2020. Across all four property types, the data predominantly reflects a positive change in prices for both categories of housing, with new detached houses being the sole exception, demonstrating a marginal decrease in average price over the specified period. Notably, new housing exhibits a notably higher rate of growth in comparison to old housing. New flats emerge as the standout category with the highest rate of growth, reaching an impressive 19%. Among old housing options, detached houses assert their resilience with a solid 6% price growth. These observed increases align closely with the potential policy effects of the First-Time Buyer's Relief, which specifically targets the market for more affordable housing options. Flats and terraced houses, typically lower-priced options, demonstrate the highest rates of price growth, further indicating how the policy might have stimulated demand and prices within these segments of the housing market.

Project 2

The Message

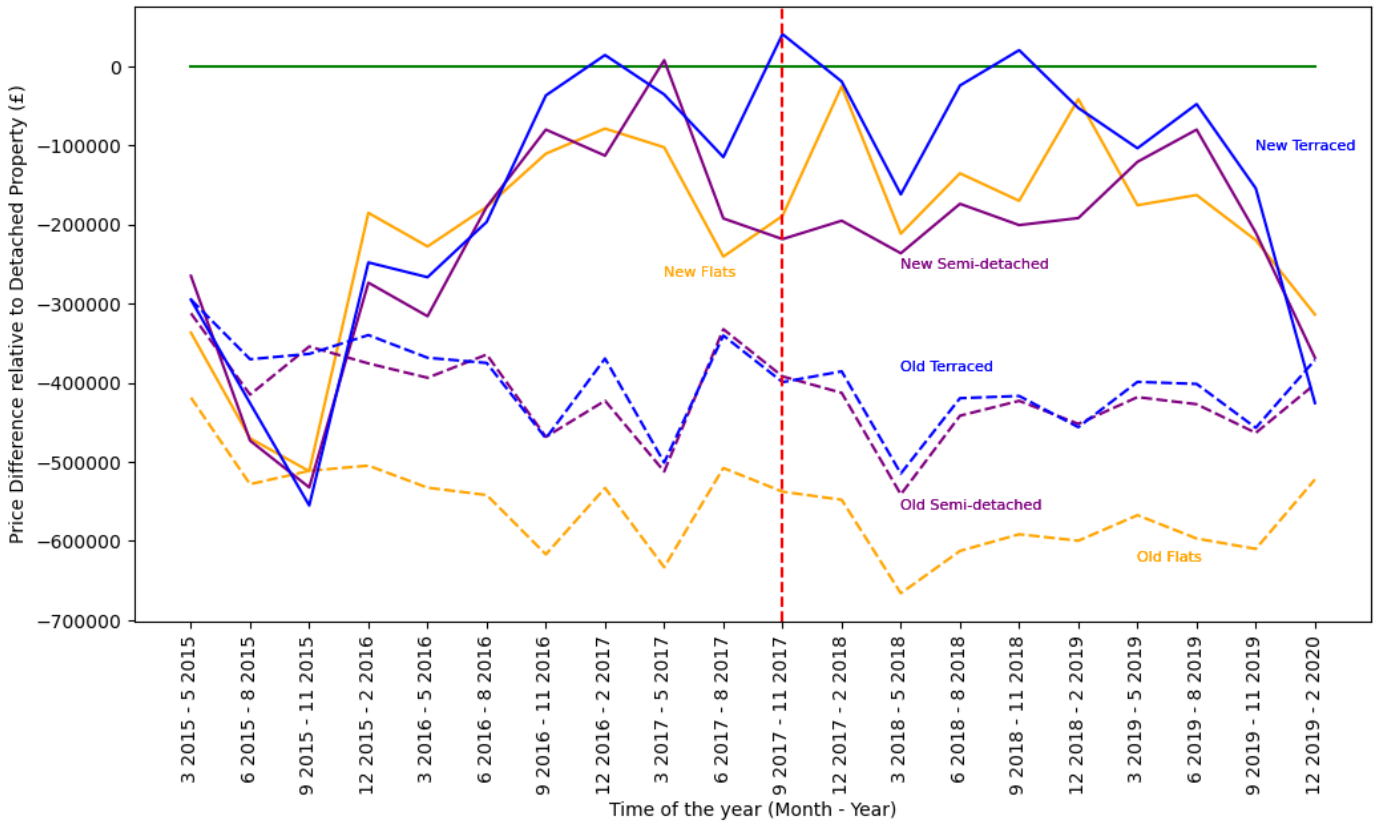
The implementation of FTB Relief on November 22, 2017 corresponds to a significant short-term price increase of purchased flats and terraced houses that were recently built in the Greater London Area, likely reflecting surge in demand if there was no significant response by an increase in housing supply in a short-term. These property types were relatively more affordable options in the market and potentially caught new buyers' attention first because they met the eligibility criteria for the tax relief.

In [29]:
```
group_means = df_copy.groupby(['Property_Type', 'Old/New', 'Interval_2'])['price'].mean(
group_means = pd.DataFrame(group_means)
group_means.reset_index(inplace=True)
```

In [30]:
```
d_properties = group_means[group_means['Property_Type'] == 'D']
filtered_group_means = group_means[(group_means['Property_Type'] != 'D') | (group_means[
# Merge the original DataFrame with the DataFrame for property type 'D' on 'Interval_2'
merged_df = pd.merge(filtered_group_means, d_properties, on=['Interval_2', 'Old/New'], s

# Calculate the price difference between each combination and store it in a new column
merged_df['price_difference'] = merged_df['price'] - merged_df['price_D']
```

In [31]:
```
# Remove rows with Property_Type 'D' and Old/New 'N' from the original DataFrame
filtered_group_means = group_means[(group_means['Property_Type'] != 'D') | (group_means[
```

In [32]:
```
import seaborn as sns

# Define the order of intervals
intervals_order = ['3 2015 - 5 2015', '6 2015 - 8 2015', '9 2015 - 11 2015',
                   '12 2015 - 2 2016', '3 2016 - 5 2016', '6 2016 - 8 2016',
                   '9 2016 - 11 2016', '12 2016 - 2 2017', '3 2017 - 5 2017',
                   '6 2017 - 8 2017', '9 2017 - 11 2017', '12 2017 - 2 2018',
                   '3 2018 - 5 2018', '6 2018 - 8 2018', '9 2018 - 11 2018',
                   '12 2018 - 2 2019', '3 2019 - 5 2019', '6 2019 - 8 2019',
                   '9 2019 - 11 2019', '12 2019 - 2 2020']

# Sort the DataFrame based on the order of intervals
merged_df_sorted = merged_df.set_index('Interval_2').loc[intervals_order].reset_index()

# Create a dictionary to map property types to labels
property_labels = {'T': 'Terraced', 'F': 'Flats', 'S': 'Semi-detached'}
old_new_4 = {'N': 'Old', 'Y': 'New'}
# Define a color palette for each property type
palette = {'T': 'blue', 'D': 'green', 'F': 'orange', 'S': 'purple'}

# Plotting the price differences for different combinations of property type and old/new
line_plot = sns.lineplot(x='Interval_2', y='price_difference', hue='Property_Type', styl

# Set the x-axis labels rotation
plt.xticks(rotation=90)

# Add a vertical line at '9 2017 - 11 2017'
plt.axvline(x='9 2017 - 11 2017', color='red', linestyle='--')

# Add title to x-axis
plt.xlabel('Time of the year (Month - Year)')
plt.ylabel('Price Difference relative to Detached Property (£)')
# Adjust figure size
plt.gcf().set_size_inches(12, 6)

line_plot.legend_.remove()


label_coordinates = {  # Specify x and y coordinates for each label
    'Terraced': {'New': ('9 2019 - 11 2019', -100000), 'Old': ('3 2018 - 5 2018', -38000
    'Detached': {'New': ('6 2015 - 8 2015', 650000), 'Old': ('9 2019 - 11 2019', -750000
    'Flats': {'New': ('3 2017 - 5 2017', -260000), 'Old': ('3 2019 - 5 2019', -620000)},
    'Semi-detached': {'New': ('3 2018 - 5 2018', -250000), 'Old': ('3 2018 - 5 2018', -5
}

for line, prop_type, linestyle in zip(line_plot.lines, merged_df_sorted['Property_Type']
    if prop_type in property_labels:  # Check if the property type is valid
        label_x, label_y = label_coordinates[property_labels[prop_type]][old_new_4[lines
        plt.text(label_x, label_y, old_new_4[linestyle] + ' ' + property_labels[prop_typ
        if linestyle == 'N':
            plt.text(label_x, label_y, '', color=palette[prop_type], horizontalalignment

plt.show()
```



This graph illustrates the price difference between detached houses and various property types over three-month intervals. However, this graph takes into account whether the property was constructed recently. Detached properties exhibited minimal price difference post-policy implementation. Therefore, the graph compares the average price of each property type to detached houses. The green line on the graph represents detached property, which indicates no change. The changes for price differences between detached property and other types around November 2017 reflect the nominal impact of the policy.

Examining the graph section marked by the red line indicating November 22nd, 2017, the day of FTB Relief implementation, reveals significant changes in trends for new terraced houses and new flats, even though they were quite consistent with other types before. These types of property experienced notable price increases compared to detached houses, with new flats almost matching and new terraced houses briefly surpassing detached house prices during that interval. The trend continued for the next three months, but then the market adjusted after the shock. These property types, meeting relief requirements, likely became highly appealing options for first-home purchases, driving demand. However, this surge was relatively brief, indicative of market adjustments coinciding with seasonal patterns, particularly as the 'cold season' started and relocation activity slowed. This paper conducts further analysis to determine the long-term effect on prices.

Maps and Interpretations

In [33]:
```
import geopandas as gpd

file_path = "/Users/user/Desktop/ECO225/ECO225Project/Data/London-wards-2018/London-ward
district_df = gpd.read_file(file_path)
```

In [34]:
```
df_maps = df_copy.groupby(['Before/After', 'District'])['price'].mean()
df_maps = df_maps.reset_index()
df_maps = df_maps.rename(columns={'price': 'Price_Mean', 'District': 'DISTRICT'})
```

In [35]:
```
#average price change for map 2
interval1 = 'Before'
interval2 = 'After'
df_interval1 = df_maps[df_maps['Before/After'] == interval1]
df_interval2 = df_maps[df_maps['Before/After'] == interval2]

# Merge the two DataFrames based on the 'DISTRICT' column
merged_df_2 = pd.merge(df_interval1, df_interval2, on='DISTRICT', suffixes=('_interval1'

# Calculate the percentage change in price
merged_df_2['Percentage_Change'] = ((merged_df_2['Price_Mean_interval2'] -
                                    merged_df_2['Price_Mean_interval1']) / merged_df_2['P

# Create a new DataFrame with 'DISTRICT' and 'Percentage_Change' columns
percentage_change_df = merged_df_2[['DISTRICT', 'Percentage_Change']]
```

In [36]:
```
district_df["DISTRICT"] = district_df["DISTRICT"].str.title().str.strip()

percentage_change_df.loc[:, "DISTRICT"] = percentage_change_df["DISTRICT"].str.title().s
```

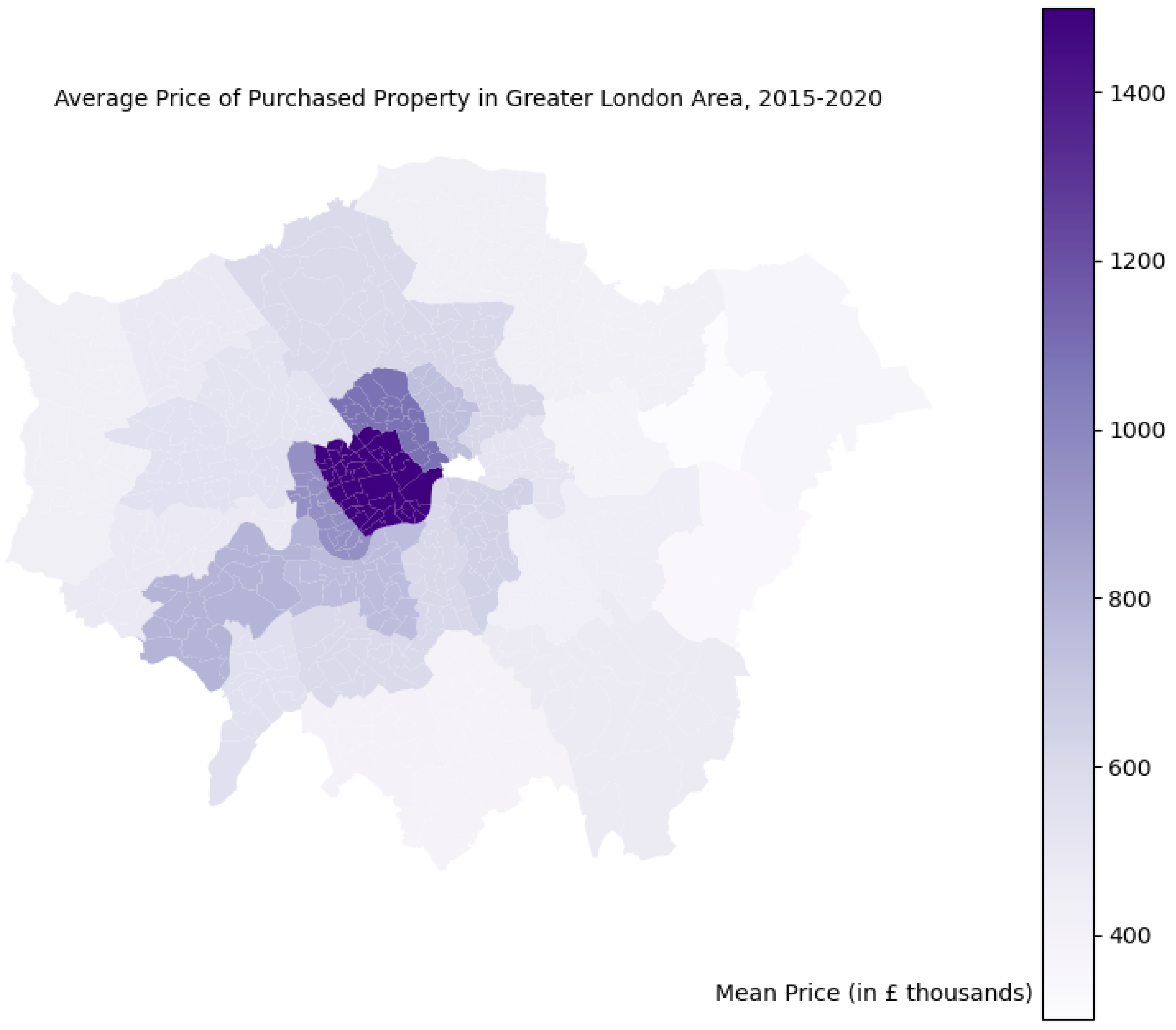
In [37]:
```
district_df['DISTRICT'].replace({'City And County Of The City Of London': 'City of Londo
```

In [38]:
```
districts = district_df.merge(percentage_change_df, left_on="DISTRICT", right_on="DISTRI
```

In [39]:
```
#map 1, average price of property in london district
grouped_district_mean = df_copy.groupby('District')['price'].mean().reset_index()
grouped_district_mean = grouped_district_mean.rename(columns={'price': 'Price_Mean', 'Di

# Convert 'price' column to mean integers and then divide by 1000
grouped_district_mean['Price_Mean'] = (grouped_district_mean['Price_Mean'].astype(int) /
grouped_district_mean = grouped_district_mean.rename(columns={'Price_Mean': 'Price_Mean
```

In [40]:
```
grouped_district_mean["DISTRICT"] = grouped_district_mean["DISTRICT"].str.title().str.st
districts_price = district_df.merge(grouped_district_mean, left_on="DISTRICT", right_on=
```

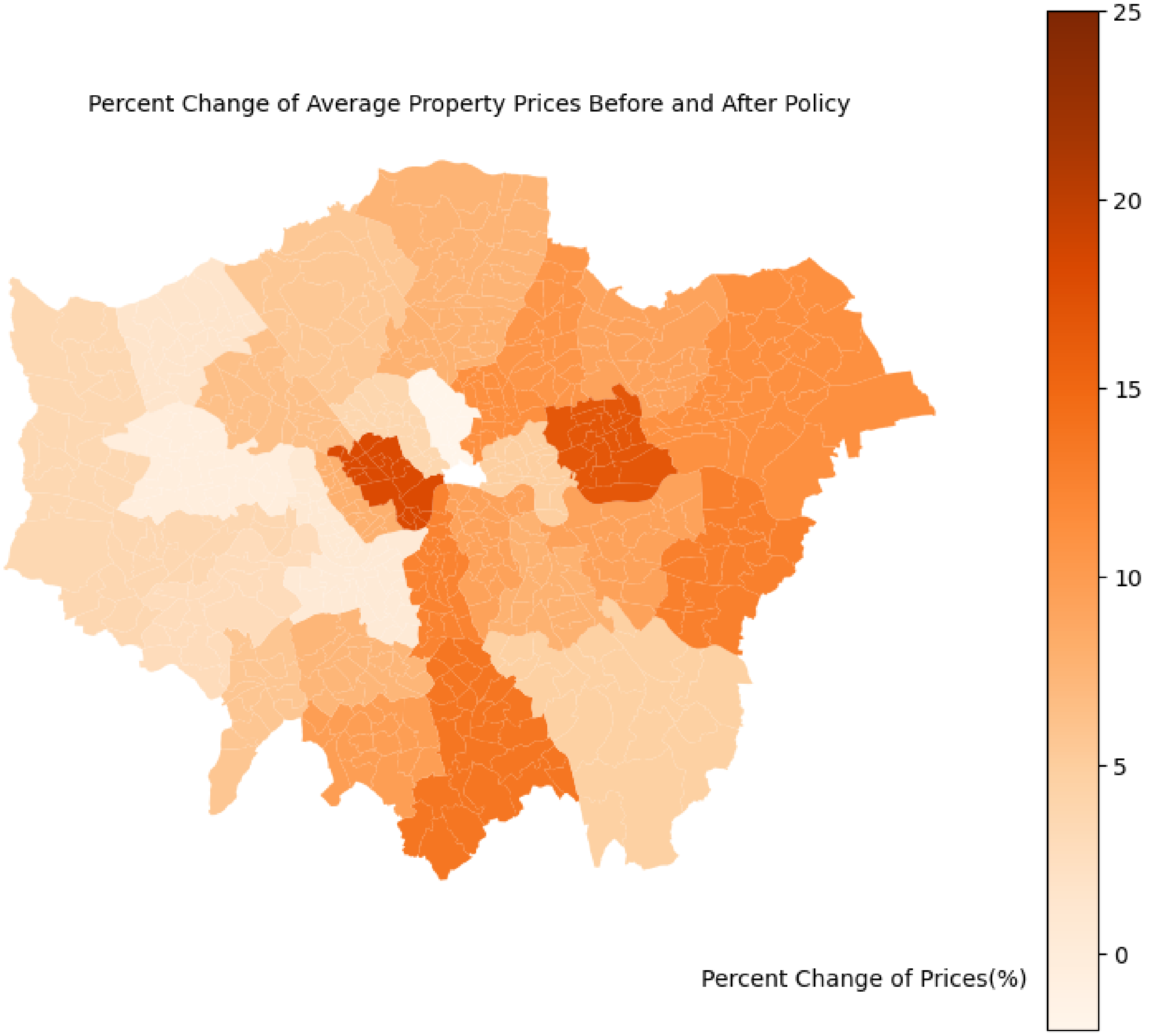
In [41]:
```
fig, gax = plt.subplots(figsize=(10, 8))

district_df.plot(ax=gax, edgecolor='white', color='white')

districts_price.plot(
    ax=gax, edgecolor='black', linewidth=0, column='Price_Mean (in thousands)', legend=T
    vmin=300, vmax=1500
)


gax.annotate('Mean Price (in £ thousands)', xy=(0.47, 0.07), xycoords='figure fraction')
gax.set_title("Average Price of Purchased Property in Greater London Area, 2015-2020", f
plt.axis('off')
plt.show()
```


Average Price of Purchased Property in Greater London Area, 2015-2020
1400
1200
1000
800
600
400
Mean Price (in £ thousands)

This map illustrates the average prices of purchased properties in each district within the Greater London Area, revealing distinct divisions in housing costs. East London regions typically offer more affordable housing options, while districts in the west tend to be more expensive. East London has some of the most low-income districts with high degrees of deprivation (Lawrence, 2022). These pricing disparities remained consistent throughout the five-year period under study. Areas such as the City of Westminster, Chelsea, and Camden, traditionally exclusive areas with lots of parks, cultural events and restaurants, stood out as the most desirable and priciest locations in London, with average property prices reaching up to 1.5 million (Manton, 2023). As the policy did not target this high-end sector of the market, there should be no significant price changes observed in these areas. In the east of London, housing tends to be more affordable, presenting potential opportunities for first-time homebuyers under the FTB Relief scheme, which could potentially cause increase in demand for housing within those districts and therefore in the housing prices. The values for the City of London borough are missing on all maps because it is not a residential district.

In [42]:
```
fig, gax = plt.subplots(figsize=(10, 8))

district_df.plot(ax=gax, edgecolor='white', color='white')
```

```
districts.plot(
    ax=gax, edgecolor='black', linewidth=0, column='Percentage_Change', legend=True, cma
    vmin=-2, vmax=25
)


gax.annotate('Percent Change of Prices(%)', xy=(0.46, 0.08), xycoords='figure fraction')
gax.set_title("Percent Change of Average Property Prices Before and After Policy", fonts
plt.axis('off')
plt.show()
```


Percent Change of Average Property Prices Before and After Policy
Percent Change of Prices(%)
25
20
15
10
5
0

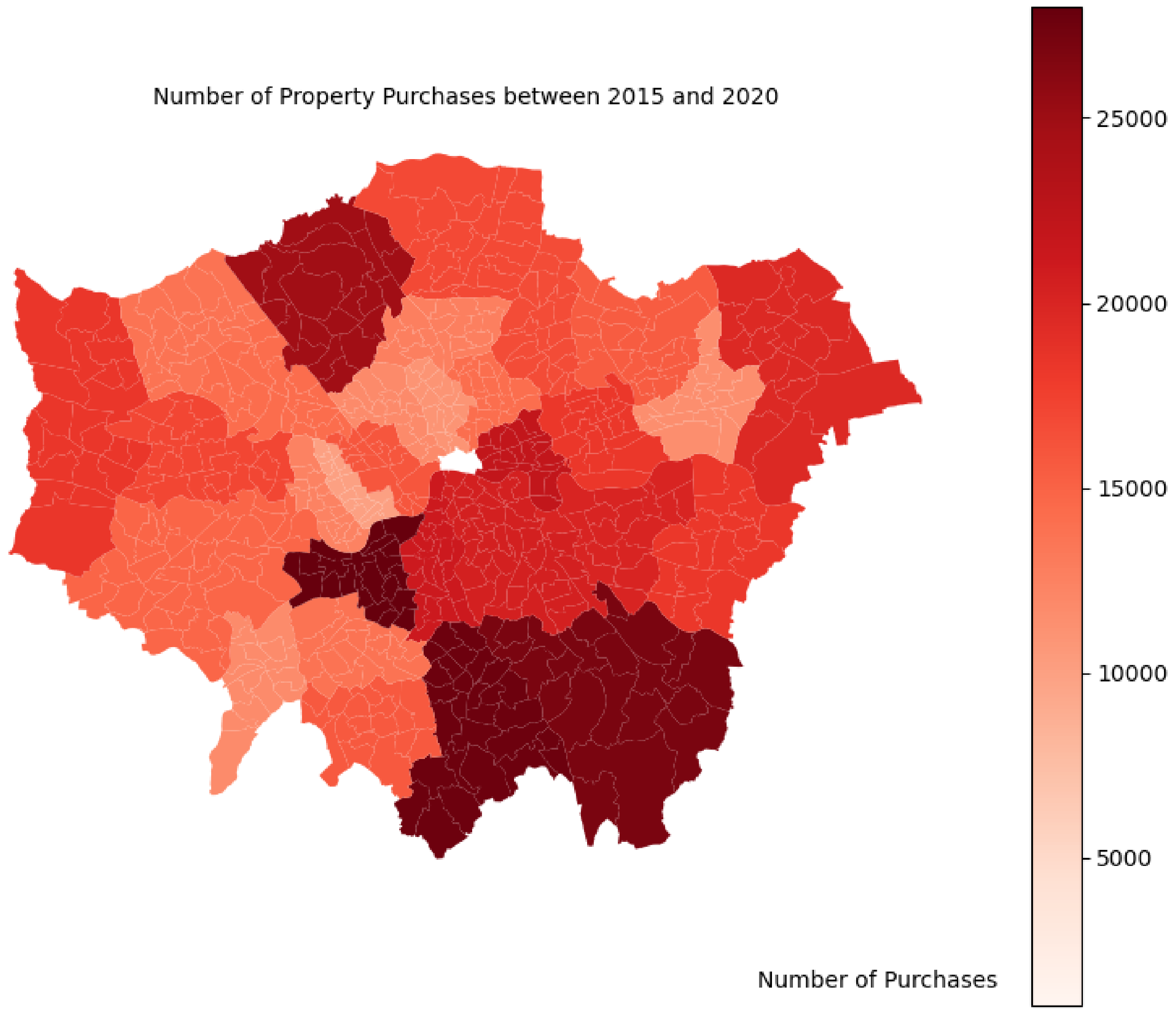
This map highlights Hammersmith and Fulham and Newham as the London districts where the value of purchased properties surged by nearly 20% before and after the policy implementation. However, the average price of property in Hammersmith and Fulham is way above threshold indicated by the UK Government for first-time buyers to qualify for the tax relief, but it is a popular area for young individuals and families (Masey, 2019).

In Newham, however, determining whether this spike can be solely attributed to increased demand resulting from the FTB Relief policy is challenging due to ongoing gentrification process (Guardian readers & Perry, 2016). Many individuals residing in more central London areas opt to relocate to Newham or nearby areas because it offers the most affordable housing while still being close to the city center. Increased housing demand might have led to price hikes in the most boroughs of East London.

```
In [43]: number_of_purchases_per_district = df_copy.groupby('District').size()

         number_of_purchases = pd.DataFrame({
             'DISTRICT': number_of_purchases_per_district.index,
             'Number of Purchases': number_of_purchases_per_district.values
         })
```

```
In [44]: number_of_purchases["DISTRICT"] = number_of_purchases["DISTRICT"].str.title().str.strip(
         districts_number = district_df.merge(number_of_purchases, left_on="DISTRICT", right_on="
```

```
In [45]: fig, gax = plt.subplots(figsize=(10, 8))

         districts_number.plot(
             ax=gax, edgecolor='black', linewidth=0, column='Number of Purchases', legend=True, c
             vmin=1000, vmax=28000
         )

         gax.annotate('Number of Purchases', xy=(0.5, 0.07), xycoords='figure fraction')
         gax.set_title("Number of Property Purchases between 2015 and 2020", fontsize=10)
         plt.axis('off')
         plt.show()
```



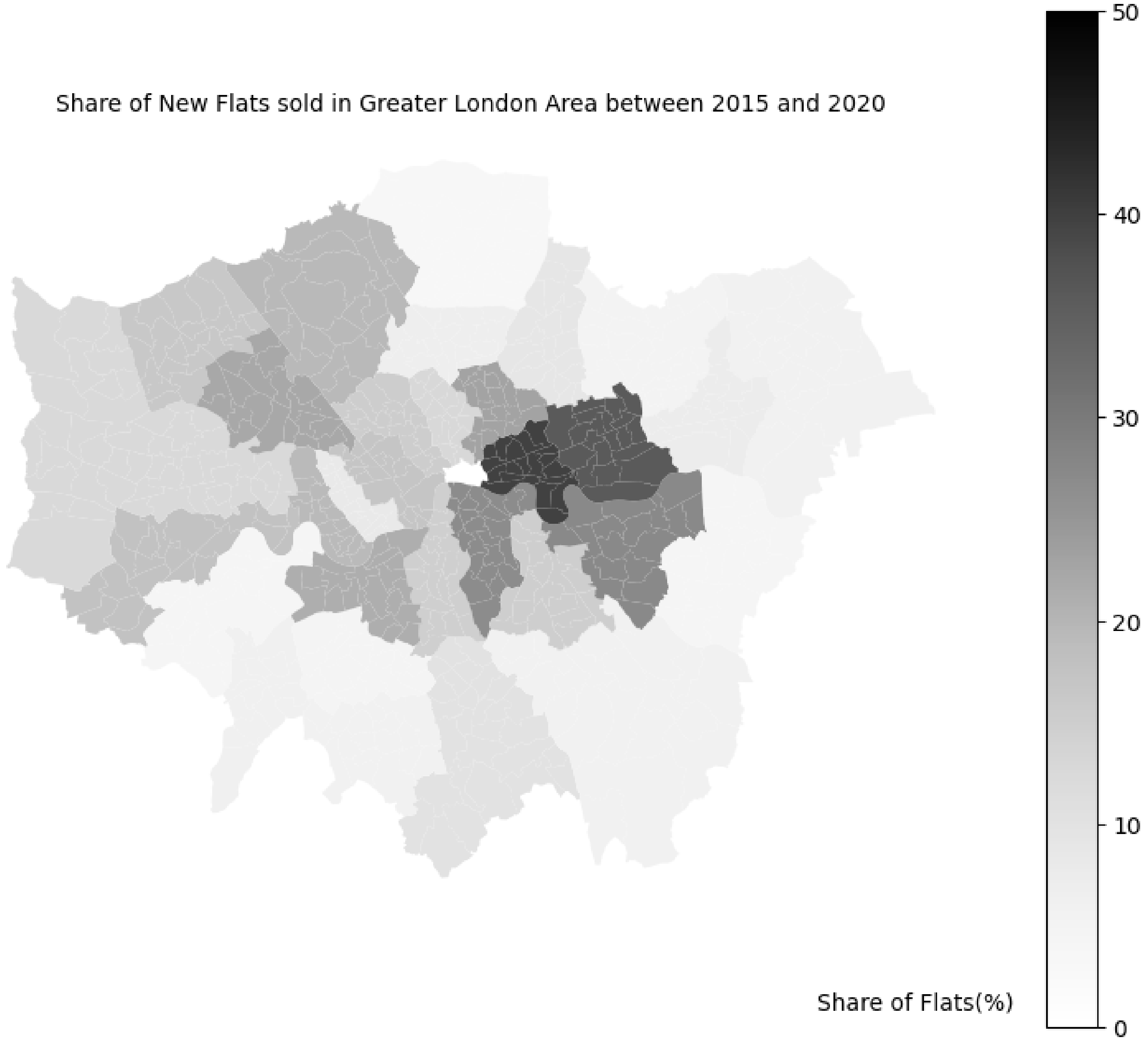
The map illustrates the distribution of property purchases across various boroughs in London. The majority of purchases occurred in three peripheral districts of London and one district within the central area. Croydon, Bromley, and Barnet, situated in the outskirts, are suburban areas known for attracting residents seeking homeownership while maintaining proximity to the city. Wandsworth, located in inner London, stands out as the most sought-after residential area (Gulliver, 2022).

Notably, Newham experienced a significant increase in property prices following the implementation of FTB Relief, despite not leading in the total number of property transactions between March 2015 and March 2020. While the volume of purchases in other boroughs remained relatively similar, districts in East London observed a more pronounced percentage change in prices.

In [46]:
```
total_purchases_district = df_copy.groupby('District').size()

new_flats_district_df = df_copy[(df_copy['Property_Type'] == 'F') & (df_copy['Old/New']

percentage_flat_district = (new_flats_district_df.groupby('District').size() / total_pur

percentage_flat_district_df = pd.DataFrame({
    'DISTRICT': percentage_flat_district.index,
    'Share of New Flats': percentage_flat_district.values
})
```

In [47]:
```
percentage_flat_district_df["DISTRICT"] = percentage_change_df["DISTRICT"].str.title().s

districts_flats = district_df.merge(percentage_flat_district_df, left_on="DISTRICT", rig
```

In [48]:
```
fig, gax = plt.subplots(figsize=(10, 8))

district_df.plot(ax=gax, edgecolor='white', color='white')

districts_flats.plot(
    ax=gax, edgecolor='black', linewidth=0, column='Share of New Flats', legend=True, cm
    vmin=0, vmax=50
)

gax.annotate('Share of Flats(%)', xy=(0.53, 0.07), xycoords='figure fraction')
gax.set_title("Share of New Flats sold in Greater London Area between 2015 and 2020", fo
plt.axis('off')
plt.show()
```


Share of New Flats sold in Greater London Area between 2015 and 2020
Share of Flats(%)
50
40
30
20
10
0

This map illustrates the prevalence of new flat purchases in inner London, reflecting the dense population in these areas. Newham and Tower Hamlets are boroughs with the biggest share of purchased new flats among all purchases in that area even though the number of purchases is similar among all boroughs.

Given that the FTB Relief policy targets lower-priced and smaller properties, flats emerge as a natural preference. Newham and Tower Hamlets are also relatively more affordable, and Newham experienced a significant price spike. All these factors likely indicate that the tax relief affected the Newham property market more than other areas, as it satisfies all requirements of the policy target. More investigation is required to understand the dynamics behind the property market of Tower Hamlets.

Project Three

Original Data Set

The original data set from HM Land Registry, the non-ministerial department that monitors and documents every housing transaction in England and Wales and has an open database for all purchases since 1995 (HM Land Registry Open Data, 2023). To begin with, each transaction had the information about where the house is located, time of transaction, and some other information, such as type of property and type of tenure. The research investigates the effect of FTB Relief, so different time variables were created and used for graphs, such as whether the purchase happened before or after the policy, intervals with duration of three and four months.

However, this data set considers mainly the demand side of purchases, ignoring the supply side. There is also no information on the area, where each house is situated. Therefore, getting more information on differences between each borough might help understand why some property markets were affected by the FTB Relief more than the others. more houses are purchased in certain places and also why some houses are on average more expensive.

Potential Data to Scrape

The data set about urban green spaces can add a lot of additional information to this research and it is not available for a download as a file. Many economists and experts on property discuss how having green spaces in the area adds 'park premium' to the average price of the house, making them more expensive (Harper, 2019). Number of green spaces near the house could have affected to what extent there was a change in demand for housing in certain boroughs after the FTB Relief implementation. This additional data set helps analyse how the number of green spaces correlates with the average housing price and number of purchases in each borough, as more families and young parents are trying to purchase a house or a flat near parks or gardens. Areas with many parks tend to be relatively more expensive, therefore prices should not increase as much, as the tax relief did not target luxurious housing markets.

The data on parks can be scraped from the Wikipedia table based on the register of Historic England and its National Heritage List about listed parks and gardens in Greater London (Registered Parks & Gardens | Historic England, n.d.).

The scraped data set will have the number of green spaces for each borough and will be merged with the original data set on the level of an individual house. Each observation will include the price of the house, its features, conditions of a purchase, location, number of parks in the area and the time of transaction.

Potential Challenges

The primary challenge is the way the data is represented because there are multiple tables. Each table is about an individual borough and has data for every park in that area. However, a borough is not mentioned in any column of the table, so the borough name should be scraped seperately.

Some boroughs have also no table because there are no registered parks or gardens in that area, which can lead to missing values in the web scraped data set.

Another challenge in using data from this Wikipedia page is that the scapped data set has each individual park and its location, whereas the total number of parks for each borough is more useful for this analysis and the future merge because all analysis are conducted on the borough level.

Some borough names differ among the original data and web scraped data, which should be taken into account.

Scraping Data from a Website

In [49]:
```
import requests
```

In [50]:
```
from bs4 import BeautifulSoup
```

Parks

In [51]:
```
web_url = 'https://en.wikipedia.org/wiki/Listed_parks_and_gardens_in_Greater_London'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
response_p = requests.get(web_url, headers=headers)
```

In [52]:
```
# We need to find all tables with format RM_table beccause that is where the information

soup_object_p = BeautifulSoup(response_p.content)
data_table_parks = soup_object_p.find_all('table', 'RM_table')
```

In [53]:
```
# Create an empty dataframe with columns of the tables

parks_df = pd.DataFrame(columns = ['borough','name', 'location', 'type', 'completed', 'g
```

To web scrape the data about parks, I manually created a list of boroughs, where the order of boroughs is the same as the order of the tables for each borough on the Wikipedia page.

In [54]:
```
boroughs = ['Barnet', 'Bexley', 'Brent', 'Bromley', 'Camden', 'City of London', 'Croydon
            'Ealing', 'Enfield', 'Greenwich', 'Hackney', 'Hammersmith and Fulham', 'Hari
            'Harrow', 'Havering', 'Hillingdon', 'Hounslow', 'Islington', 'Kensington and
            'Lambeth', 'Lewisham', 'Merton', 'Newham', 'Redbridge', 'Richmond upon Thame
            'Sutton', 'Tower Hamlets', 'Wandsworth', 'Westminster']
```

Then, 'while' and 'for' loops are used to create one large data frame, where the order of boroughs in the list above is used to indicate where each park is located. The index variable *i* is set to 0. The loop continues until *i* reaches 30, which is the total number of parks.

The 'tr' tag finds an individual table on the website, and all parks in that borough are included in 'all_values_parks'. The 'borough' is determined based on the number of the table.

Then, the 'for' loop treats each row as a potential park, where information about an individual park is extracted using the 'td' function of Beautiful Soup. The loop seperates a name, a location, a type of green space, a completion date, and coordinates from the values based on their index. All information is then added to the data frame _parks*df*.

In [55]:
```
i = 0
while i < 30:
    all_values_parks = data_table_parks[i].find_all('tr')
    borough = boroughs[i]
    for row in all_values_parks[1:]:
        values = row.find_all('td')
        name = values[5].text.strip()
        location = values[0].text
        type1 = values[1].text
        completed = values[2].text
        coord = values[3].text.strip()

        # Append a new row to the DataFrame using .loc
        parks_df.loc[len(parks_df)] = [borough, name, location, type1, completed, coord]

    i = i + 1
```

Groupby and size fuctons are used to find the total number of parks in each borough. Waltham Forest, Kingston Upon Thames and Barking And Dagenham are added seperately manually, as these boroughs do not have any registered parks or gardens. They are not listed on the web page, but missing values would disrupt the merge process.

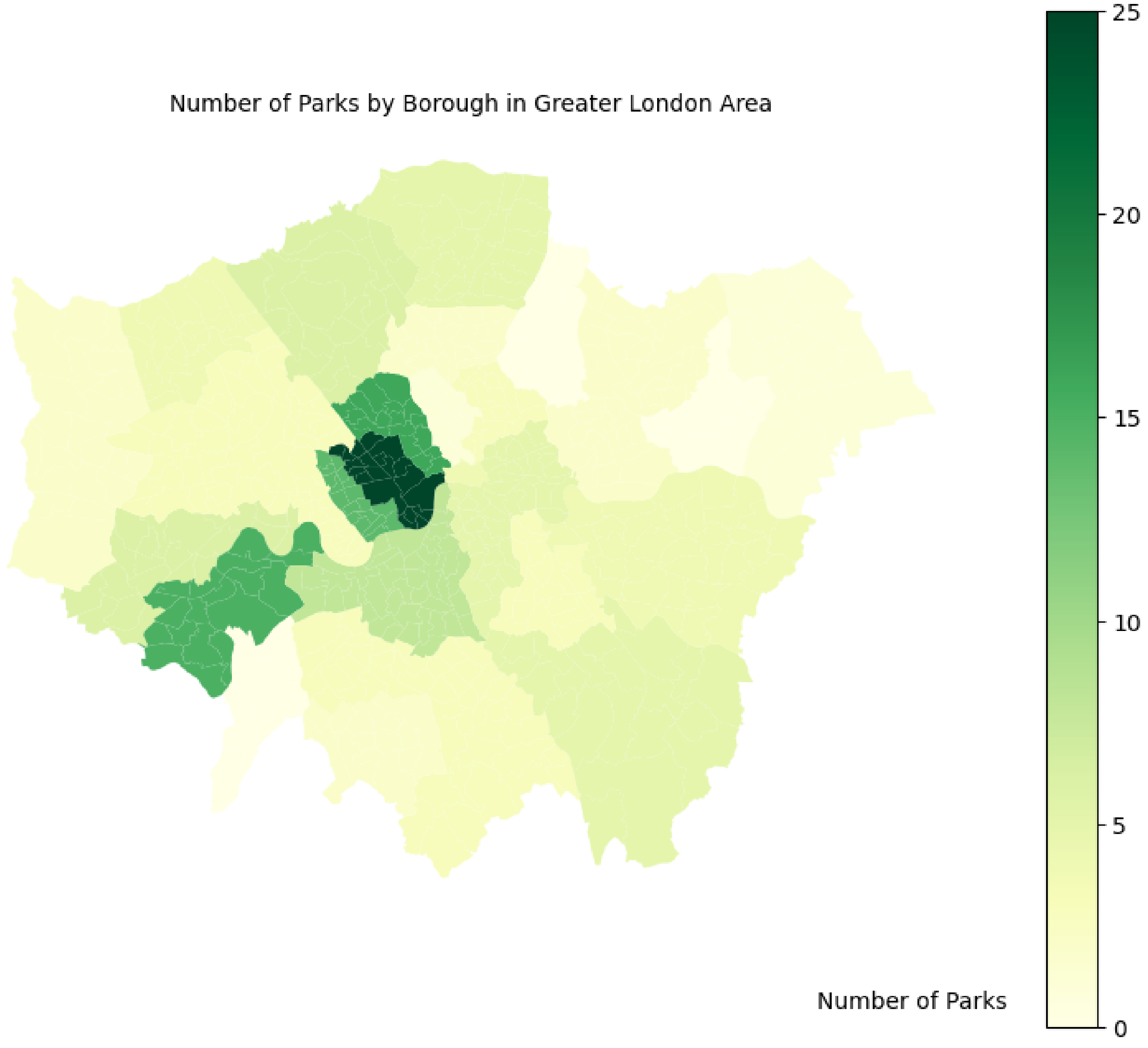
In [56]:
```
# Calculate total numbr of parks for each borough, some boroughs do not have any parks
parks_by_borough = parks_df.groupby('borough').size().reset_index(name='num_parks')
parks_by_borough["borough"] = parks_by_borough["borough"].str.title().str.strip()
districts_with_zero_parks = ['Waltham Forest', 'Kingston Upon Thames', 'Barking And Dage

# Create a DataFrame for districts with 0 parks
districts_with_zero_parks_df = pd.DataFrame({'borough': districts_with_zero_parks, 'num_

# Concatenate the two DataFrames
parks_by_borough_new = pd.concat([parks_by_borough, districts_with_zero_parks_df], ignor
```

The name of Westminster borough is changed to 'City Of Westminster', as it is the name used in the other data sets. Then, I merge the new data set with the old one on the level of each individual transaction, where _num*parks* indicates number of green spaces in the area where the house was purchased.

In [57]:
```
district_df["DISTRICT"] = district_df["DISTRICT"].str.title().str.strip()
parks_by_borough_new["borough"] = parks_by_borough_new["borough"].str.title().str.strip(
parks_by_borough_new['borough'] = parks_by_borough_new['borough'].replace('Westminster',
borough_parks = district_df.merge(parks_by_borough_new, left_on="DISTRICT", right_on="bo
```

Visualizing the Scraped Data Set

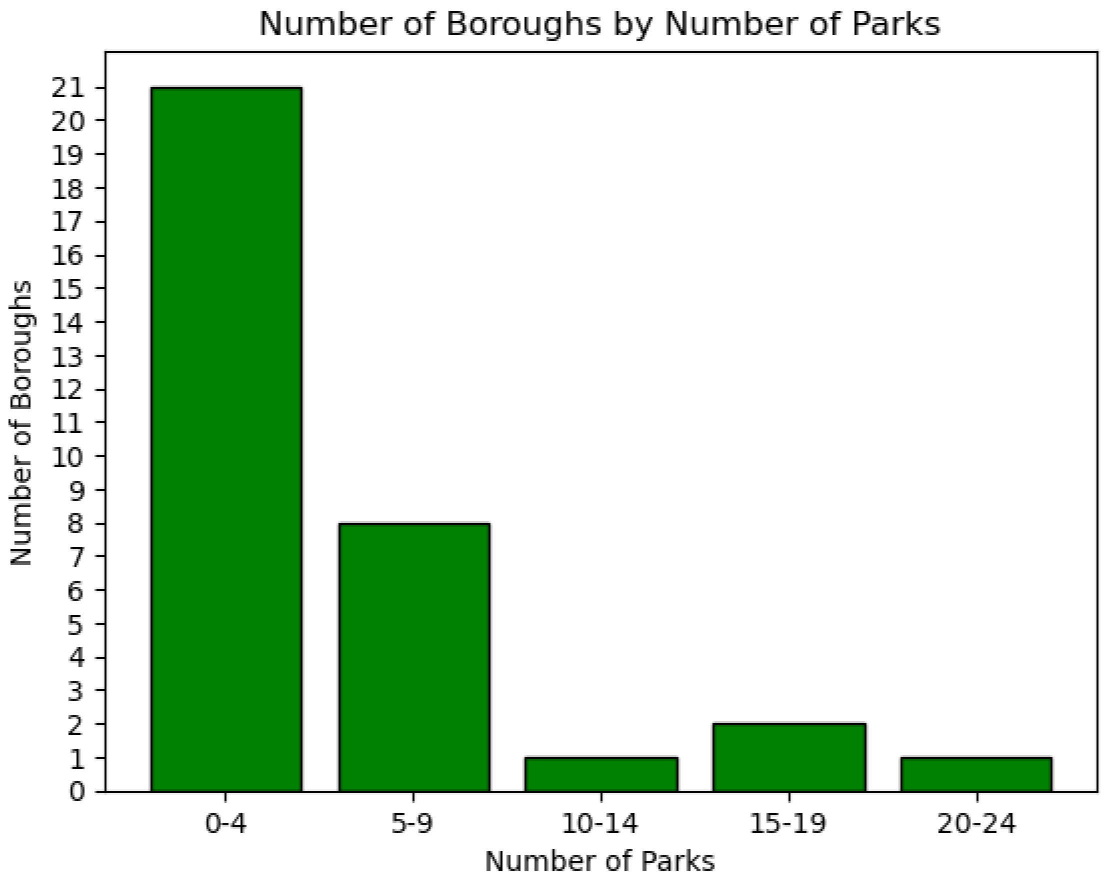
In [58]:
```
fig, gax = plt.subplots(figsize=(10, 8))

district_df.plot(ax=gax, edgecolor='white', color='white')
borough_parks.plot(
    ax=gax, edgecolor='black', linewidth=0, column='num_parks', legend=True, cmap='YlGn'
    vmin=0, vmax=25
)



gax.annotate('Number of Parks', xy=(0.53, 0.07), xycoords='figure fraction')
gax.set_title("Number of Parks by Borough in Greater London Area", fontsize=10)
plt.axis('off')
plt.show()
```



Considering previous analysis of the map with an average price for each borough, this map highlights an economic trend in West London. It shows that boroughs with extremely large number of parks were also the same boroughs that experienced minimal price increases before and after the implementation of first-time buyer policies, yet maintained the highest average prices.

This suggests that homes near parks tend to command a premium, emphasizing the role of amenities in driving property values. These areas did not experience significant changes in the context of policy interventions such as first-time buyer initiatives, which underscores the relevance of urban features like proximity of green spaces in shaping housing market dynamics.

In [59]:
```
boroughs = parks_by_borough_new['borough']
num_parks = parks_by_borough_new['num_parks']

# Define bins
bins = [0, 5, 10, 15, 20, 25]

# Count the number of boroughs falling within each bin
hist, _ = np.histogram(num_parks, bins=bins)

# Plot the bar chart
plt.bar(range(len(hist)), hist, align='center', color='green', edgecolor='black')
plt.xticks(range(len(bins) - 1), [f"{bins[i]}-{bins[i+1]-1}" for i in range(len(bins) -
plt.xlabel('Number of Parks')
plt.ylabel('Number of Boroughs')
plt.title('Number of Boroughs by Number of Parks')

# Set y-axis ticks to integers only
plt.yticks(np.arange(max(hist) + 1))

plt.show()
```



The distribution of parks across Greater London boroughs underscores a significant economic phenomenon. Most boroughs exhibit a scarcity of parks, typically numbering between 0 to 9 within their boundaries, with only a few exceptions boasting a higher concentration of green spaces. The biggest share of purchased property before and after the policy implementation is located in boroughs that fall within the first category, therefore FTB Relief should affect areas with less parks more.

In [60]:
```
df_copy["District"] = df_copy["District"].str.title().str.strip()
df_new = pd.merge(df_copy, parks_by_borough_new, left_on='District', right_on='borough',
```

The inner merge for the park data set and the original data set is done. No observations are lost.

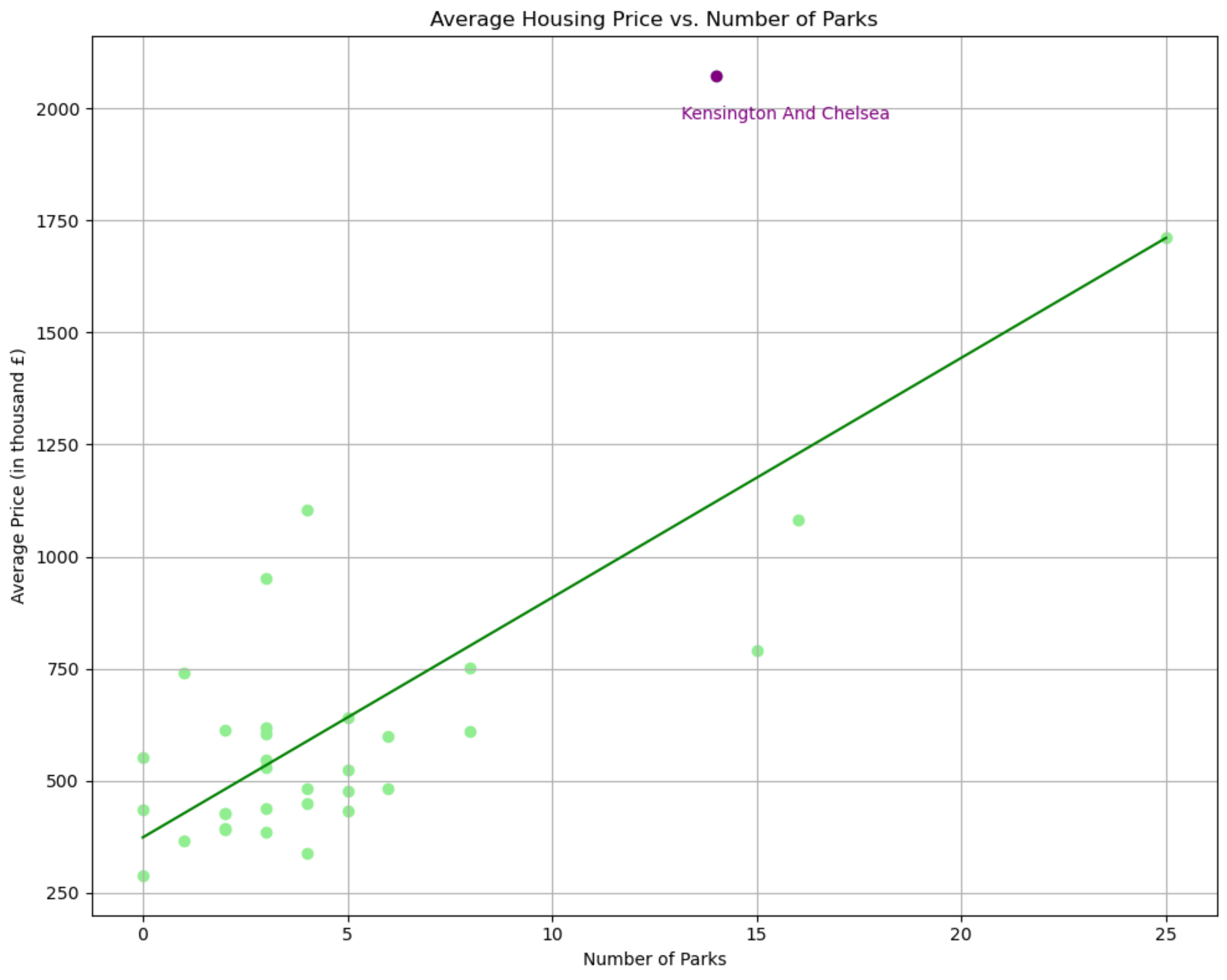
In [61]:
```
copy_3 = df_new.copy()
copy_3['price'] = copy_3['price'] / 1000
average_price_by_area = copy_3.groupby('borough')['price'].mean()
```

In [62]:
```
number_of_parks_borough = df_new.groupby('borough')['num_parks'].mean()
```

In [63]:
```
data_4 = {
    'Average Price (thousand £)': average_price_by_area,
    'Number of Parks': number_of_parks_borough
}
```

```
result_df_4 = pd.DataFrame(data_4).reset_index()
```

In [64]:
```
fig, ax = plt.subplots(figsize=(10, 8))


ax.scatter(result_df_4['Number of Parks'], result_df_4['Average Price (thousand £)'], co

# Plot Kensington and Chelsea data point separately
kensington_chelsea_index = result_df_4[result_df_4['borough'] == 'Kensington And Chelsea
ax.scatter(result_df_4.loc[kensington_chelsea_index, 'Number of Parks'],
           result_df_4.loc[kensington_chelsea_index, 'Average Price (thousand £)'],
           color='purple', label='Kensington And Chelsea')

# Plot the linear trendline
coefficients = np.polyfit(result_df_4['Number of Parks'], result_df_4['Average Price (th
trendline_x = np.array([min(result_df_4['Number of Parks']), max(result_df_4['Number of
trendline_y = np.polyval(coefficients, trendline_x)
ax.plot(trendline_x, trendline_y, color='green', label='Trendline')


ax.set_xlabel('Number of Parks')
ax.set_ylabel('Average Price (in thousand £)')
ax.set_title('Average Housing Price vs. Number of Parks')
ax.grid(True)

for i in kensington_chelsea_index:
    ax.annotate('Kensington And Chelsea',
                xy=(result_df_4.loc[i, 'Number of Parks'], result_df_4.loc[i, 'Average P
                xytext=(-20, -25), textcoords='offset points', color = 'purple')

plt.tight_layout()
plt.show()
```



Each dot on this diagram represents a borough. The moderatly strong positive relationship between park abundance and average housing prices across Greater London reveals how boroughs with a higher number of parks generally tend to command higher average property prices, Kensington and Chelsea stands out as a unique case. Despite not having the largest number of parks among boroughs, its remarkable count of 14 parks is noteworthy. Nonetheless, it remains the most expensive in terms of housing, underscoring its exceptional status as the London's most luxurious and exclusive property market (Manton, 2023).

This discrepancy highlights the potential interplay between amenities like parks and the prestigious reputation of certain areas within the city's real estate landscape. This graph indicates again that the property markets with many green spaces tend to have a 'park premium' and did not experience any sigificant changes after the policy implementation. Areas with more parks tend to not only be more expensive but have less price flactuations because supply and demand should be more inelastic, as these are historic areas and it is hard to build new houses there.

Adding a New Data Set

The new data set contains information on total affordable housing completions by financial year in each London borough since 1991/92. It includes homes funded through programs managed by the GLA (and formerly by the Homes and Communities Agency), as well as homes funded through other sources and programs. This data set defines affordable housing as the sum of social rent, affordable rent, intermediate rent, and low-cost home ownership. Additionally, it defines new affordable homes as housing units provided to specified eligible households whose needs are unmet by the market. The data is sourced from the Homes and Communities Agency and Local Authorities, providing comprehensive insights into the affordable housing landscape in London.

I merged the new data set on the level of an individual purchase. Each transaction indicates information about the property, such as age, type and tenure, as well as number of affordable houses built in the borough within the same year of transaction.

This analysis should gain a comprehensive understanding of the impact of FTB Relief on the housing market in the Greater London Area. This integration enables a thorough examination of how FTB Relief influenced housing market dynamics across various property types, considering both transactional activity from the original HM Land Registry data set and affordable housing completions from the new data set.

Importantly, the inclusion of the new data set allows for a partial observation of the supply side of the affordable housing market. This enables an analysis of to what extent the effect of FTB Relief policy on the average price of the purchased property could have been influenced by the quantity of affordable housig supplied in that area. By assessing how changes in affordable housing completions correspond to fluctuations in housing prices in different boroughs, this holistic approach provides valuable insights into the multifaceted dynamics of the housing market response to policy interventions.

Even though these are affordable housing units, landlords and authorities may increase the annual rent by the change to Consumer Price Index(CPI) plus 1 percentage point with the maximum 'ceiling' of 7% (Wilson , 2022).

Data Cleaning

The data set has four columns, including Code, Area, Year and Affordable Housing Supply. GLA uses fiscal years, therefore they align with Interval_1 variable of the original data set. However, a different formatig is used, so to change '2015-16' to 'March 2015 - February 2016', a new fuction is applied to each observation in the new data set. After this process, the name of some borough are changed to match, and then affordable housing supply observations are merged with the original data set on the level of each transaction.

In [65]:
```
second_dataset_path = "/Users/user/Desktop/ECO225/ECO225Project/Data/dclg-affordable-hou

colnames_new = ['Code', 'Area', 'Year', 'Affordable Housing Supply']

df_supply = pd.read_csv(second_dataset_path, header=None, names=colnames_new)

filtered_df_supply = df_supply[df_supply['Year'].between('2015-16', '2019-20')]
```

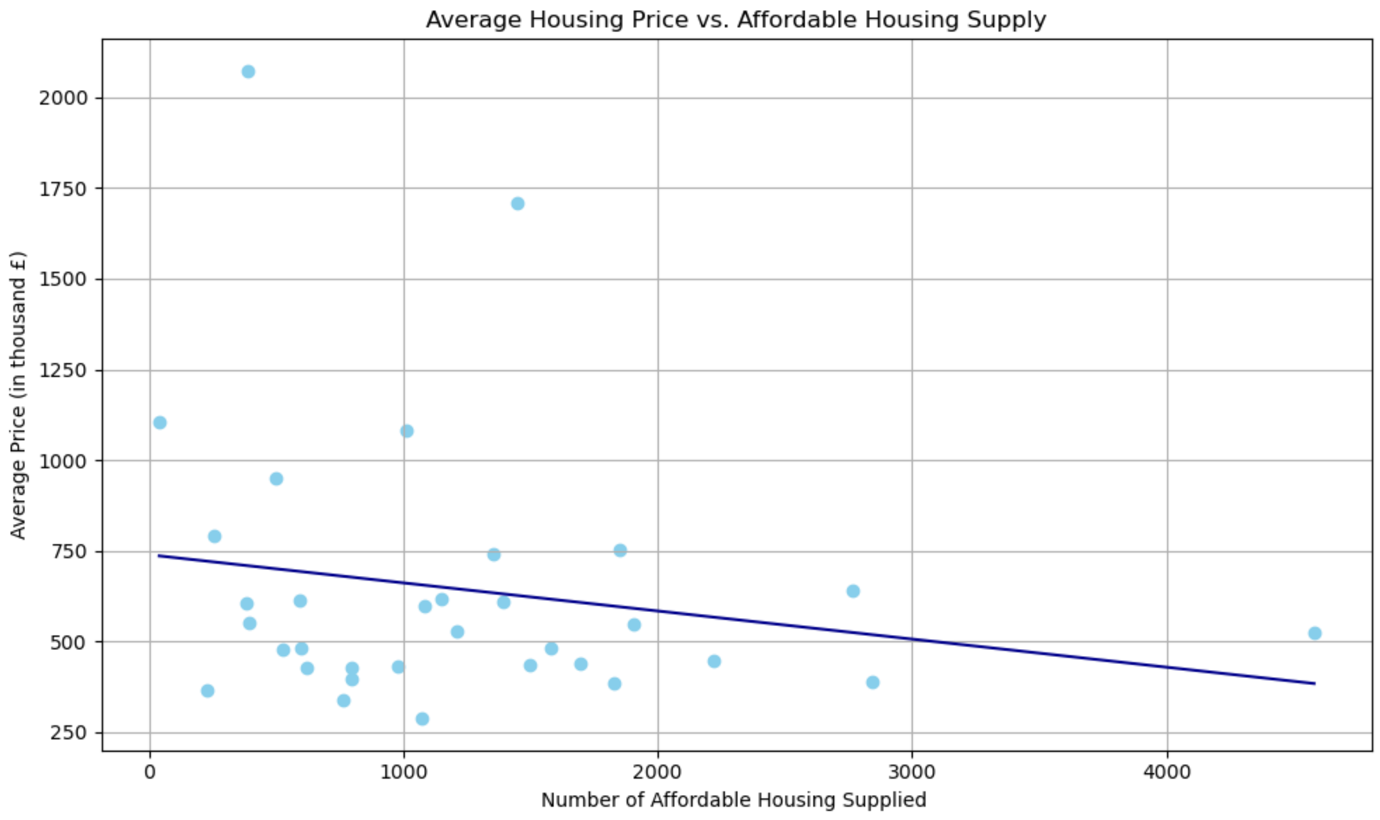
In [66]:
```
filtered_df_supply.loc[:, "Area"] = filtered_df_supply["Area"].str.title().str.strip()
```

In [67]:
```
def convert_date_range(date_range):
    parts = date_range.split('-')

    # Extract the start and end years from the date range
    start_year = parts[0]
    end_year = parts[1]
    # the end year is represented with two digits, convert to 'YYYY'
    if len(end_year) == 2:
        end_year = '20' + end_year

    # Generate formatted date range
    formatted_date_range = f"March {start_year} - February {end_year}"

    return formatted_date_range



# The conversion function to the 'Year' column using .loc[]
filtered_df_supply.loc[:, 'Year'] = filtered_df_supply['Year'].apply(lambda x: convert_d
```

In [68]:
```
filtered_df_supply_copy = filtered_df_supply.copy()
filtered_df_supply_copy.loc[filtered_df_supply_copy["Area"] == 'Westminster', "Area"] =
final_df = pd.merge(df_new, filtered_df_supply_copy, left_on=['District', 'Interval_1'],
```

In [69]:
```
final_df_copy = final_df.copy()
final_df_copy['Affordable Housing Supply'] = final_df_copy['Affordable Housing Supply'].
final_df_copy['Affordable Housing Supply'] = final_df_copy['Affordable Housing Supply'].

# Divide 'price' column by 1000

final_df_copy['price'] /= 1000

average_price_by_area = final_df_copy.groupby('Area')['price'].mean()

# Group by 'Area' and calculate the average price for each group

total_supply_by_year = final_df_copy.groupby(['Area', 'Year'])['Affordable Housing Suppl

# Reset the index to make the grouped columns accessible

total_supply_by_year = total_supply_by_year.reset_index()

# Calculate total supply by year

total_supply_by_area_2 = total_supply_by_year.groupby('Area')['Affordable Housing Supply

# Combine average price with total supply by area

data_3 = {
    'Average Price (thousand £)': average_price_by_area,
    'Affordable Housing Supply': total_supply_by_area_2
}

# Create a new DataFrame
result_df_3 = pd.DataFrame(data_3).reset_index()
```

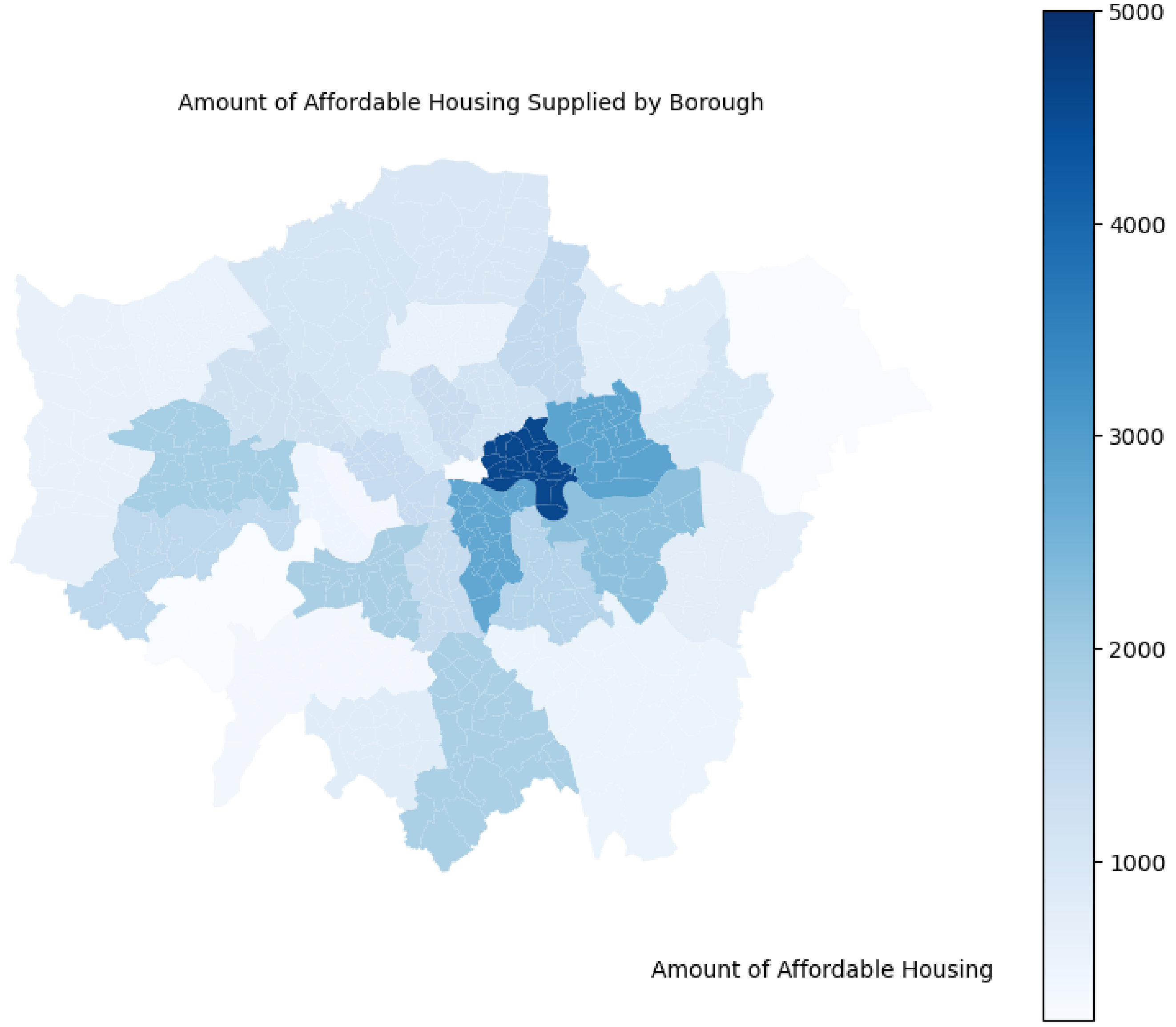
In [70]:
```
plt.figure(figsize=(10, 6))
plt.scatter(result_df_3['Affordable Housing Supply'], result_df_3['Average Price (thousa
plt.xlabel('Number of Affordable Housing Supplied ')
plt.ylabel('Average Price (in thousand £)')
plt.title('Average Housing Price vs. Affordable Housing Supply')

# Fit a linear trendline
coefficients = np.polyfit(result_df_3['Affordable Housing Supply'], result_df_3['Average
trendline_x = np.array([min(result_df_3['Affordable Housing Supply']), max(result_df_3['
trendline_y = np.polyval(coefficients, trendline_x)
plt.plot(trendline_x, trendline_y, color='darkblue', label='Trendline')

plt.grid(True)
plt.tight_layout()
plt.show()
```



The graph includes data points for each borough and highlights a clear negative relationship between the number of affordable housing units supplied in a borough and the average price of houses in that area, which is likely because more affordable housing is supplied in poorer areas. This observation holds significant policy implications, particularly considering that FTB Relief targets the lower-cost segment of the market. This graph implies that individuals should be drawn to areas with higher availability of affordable housing, as they have lower average prices, because it maximizes the benefits of the policy. Ultimately, this underscores how FTB Relief should have affected not only more affordable areas but areas with higher supply of affordable housing, which often tend to be the same.

In [71]:
```
result_df_3["Area"] = result_df_3["Area"].str.title().str.strip()
result_df_3["Area"] = result_df_3["Area"].replace('Westminster', 'City Of Westminster')
result_df_3_map = district_df.merge(result_df_3, left_on="DISTRICT", right_on="Area", ho
```

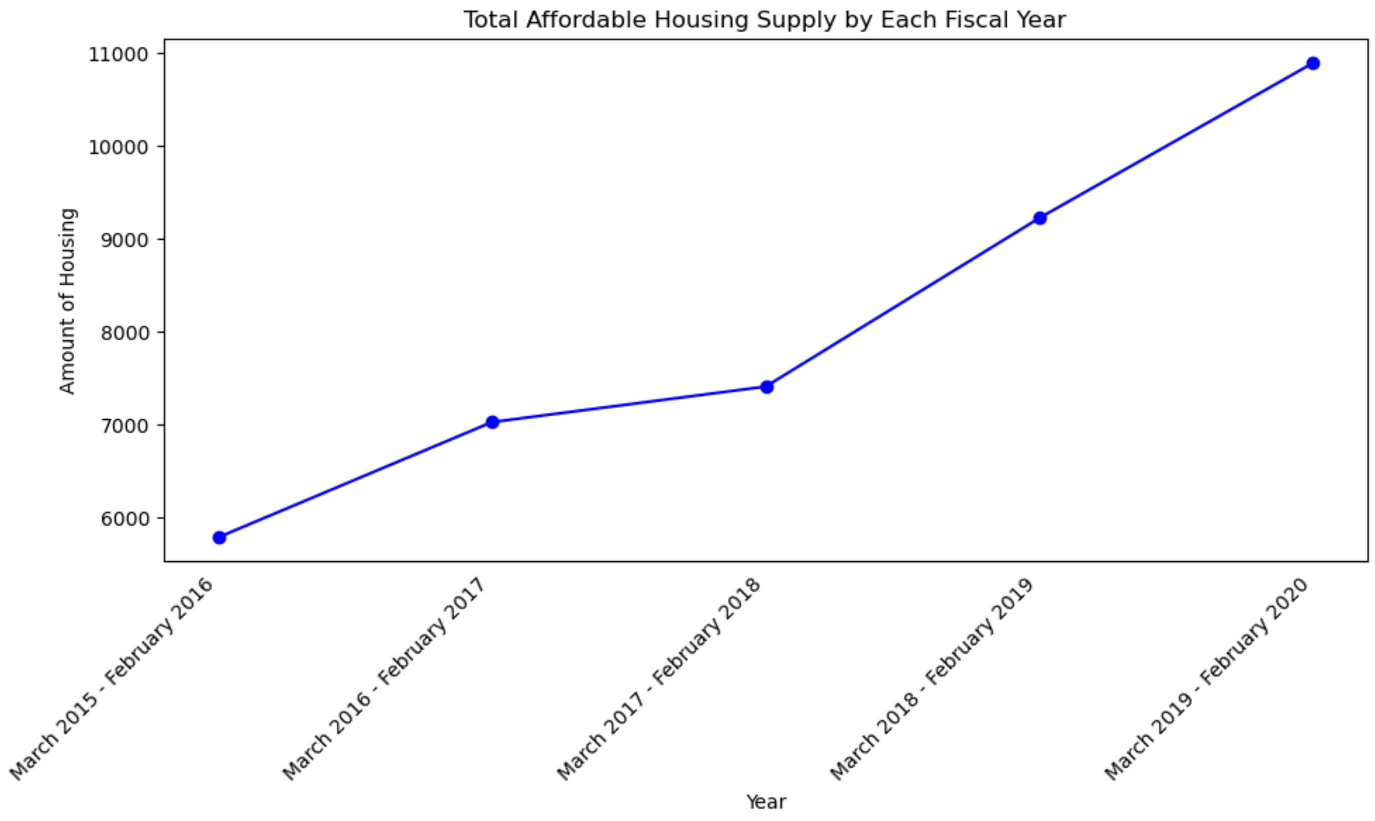
In [72]:
```
fig, gax = plt.subplots(figsize=(10, 8))

result_df_3_map.plot(
    ax=gax, edgecolor='black', linewidth=0, column='Affordable Housing Supply', legend=T
    vmin=250, vmax=5000
)


gax.annotate('Amount of Affordable Housing', xy=(0.43, 0.08), xycoords='figure fraction'
gax.set_title("Amount of Affordable Housing Supplied by Borough", fontsize=10)
plt.axis('off')
plt.show()
```


Amount of Affordable Housing Supplied by Borough
5000
4000
3000
2000
1000
Amount of Affordable Housing

The map highlights Tower Hamlets, Newham, and Southwark as significant hubs of affordable housing provision over five years, particularly in the context of First Time Buyers' (FTB) Relief. From the previous analysis, Tower Hamlets and Newham exhibit similar characteristics, including the highest shares of purchased new flats among all London boroughs and comparable green space amenities. However, while Newham experienced a notable price spike post-policy, Tower Hamlets saw only a minor price adjustment. This difference underscores larger affordable housing supply in Tower Hamlets, which could potentially counteract increased demand after FTB Relief, moderating price fluctuations. The contrasting outcomes highlight the importance of housing supply elasticity in shaping the economic impacts of policy interventions within urban housing markets.

In [73]:
```
total_supply_by_year_2 = total_supply_by_year.groupby('Year')['Affordable Housing Supply
```

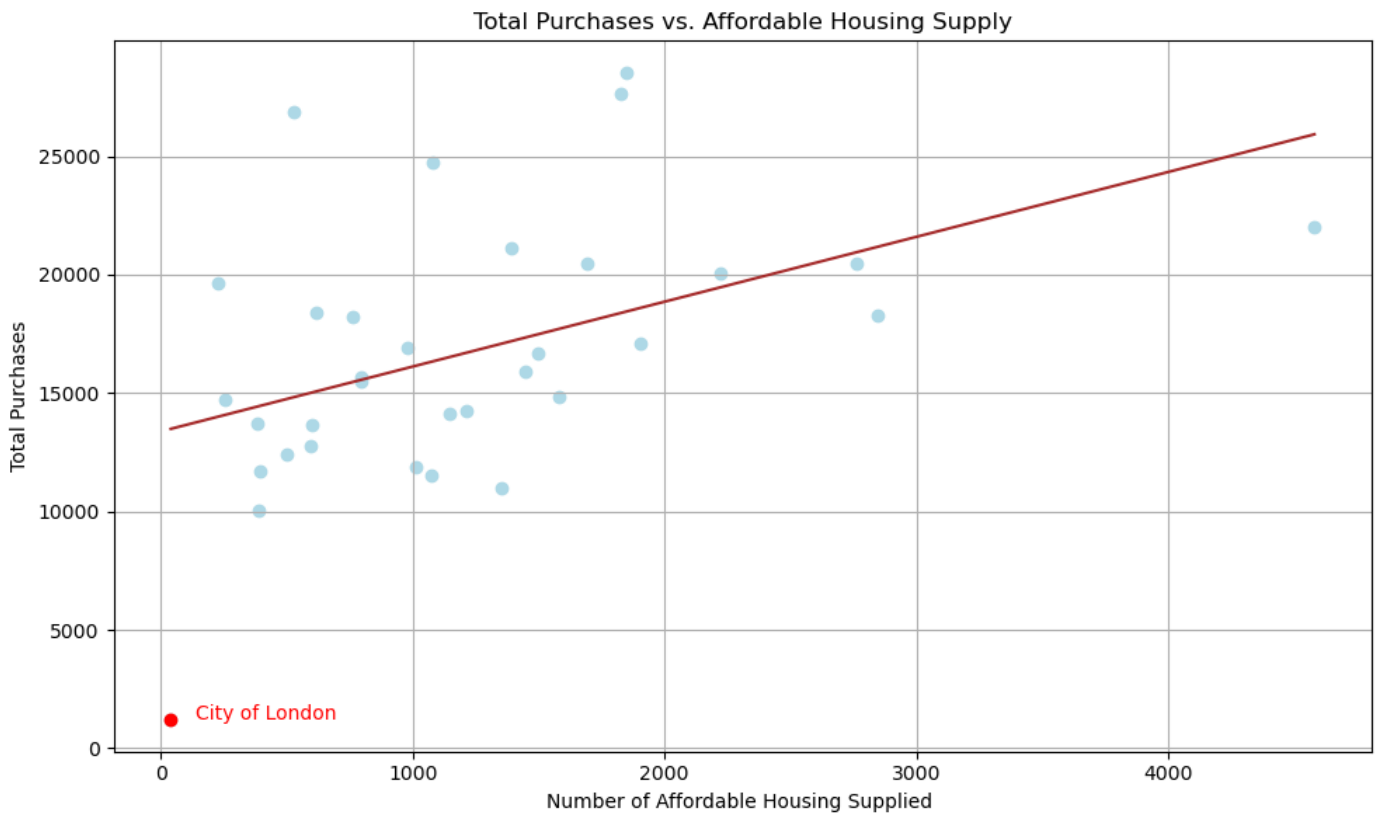
In [74]:
```
plt.figure(figsize=(10, 6))
total_supply_by_year_2.plot(marker='o', linestyle='-', color='blue')
plt.title('Total Affordable Housing Supply by Each Fiscal Year')
plt.xlabel('Year')
plt.ylabel('Amount of Housing')
plt.xticks(rotation=45, ha='right')
plt.tick_params(axis='x', which='both', bottom=False, top=False)
plt.tight_layout()
plt.show()
```



The graph depicts a significant surge in the supply of affordable housing subsequent to the implementation of First Time Buyers' (FTB) Relief in November 2017. This notable increase might signify a proactive response to the increased demand and rise in average housing prices caused by the policy intervention. The surge in affordable housing supply can highlight the responsiveness of the market to shifting demand dynamics, potentially reflecting an effort to meet the increased housing needs of first-time buyers incentivized by the policy. This dynamic might illustrate the crucial interplay between supply and demand forces within the housing market, highlighting the adaptability of the market to policy interventions aimed at enhancing housing affordability. Overall, the observed post-policy increase in affordable housing supply can underscore a strategic approach to address affordability concerns and accommodate the growing demand stimulated by FTB Relief.

In [75]:
```
total_purchases_by_area = final_df_copy.groupby('Area')['price'].count()
total_purchases_by_area  = total_purchases_by_area .reset_index()
total_purchases_by_area.rename(columns={'price': 'Total purchases'}, inplace=True)
total_purchases_by_area = pd.DataFrame(total_purchases_by_area)
result_df_5 = total_purchases_by_area.merge(result_df_3, left_on="Area", right_on="Area"
```

In [76]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(10, 6))

# Scatter plot for all boroughs
plt.scatter(result_df_5['Affordable Housing Supply'], result_df_5['Total purchases'], co

# Scatter plot for City of London separately
city_london_index = result_df_5[result_df_5['Area'] == 'City Of London'].index
plt.scatter(result_df_5.loc[city_london_index, 'Affordable Housing Supply'],
            result_df_5.loc[city_london_index, 'Total purchases'],
            color='red')

# Fit a linear trendline
coefficients = np.polyfit(result_df_5['Affordable Housing Supply'], result_df_5['Total p
trendline_x = np.array([min(result_df_5['Affordable Housing Supply']), max(result_df_5['
trendline_y = np.polyval(coefficients, trendline_x)
plt.plot(trendline_x, trendline_y, color='brown')

# Text annotation for City of London
plt.text(float(result_df_5.loc[city_london_index, 'Affordable Housing Supply'].iloc[0]),
         float(result_df_5.loc[city_london_index, 'Total purchases'].iloc[0]),
         'City of London', color='red')

plt.xlabel('Number of Affordable Housing Supplied ')
plt.ylabel('Total Purchases')
plt.title('Total Purchases vs. Affordable Housing Supply')
plt.grid(True)
plt.tight_layout()
plt.show()
```



The positive relationship observed between total house purchases and affordable housing supply in each borough reflects that the increase in affordable housing supply correlates with a rise in the quantity of purchased houses, in line with economic theory. Each dot on the graph represents a borough. City of London is a borough with mostly commercial and government buildings, therefore there are not as many property options available for purchase or supplied overall. Following the First Time Buyers’ (FTB) Relief implementation, an increase in affordable housing supply could lead to lower prices and drive increased house purchases in certain areas. This alignment with economic theory underscores how housing supply dynamics can influence consumer behavior and shape the housing market landscape.

Final Project

Adding a new data set

```
In [77]: import pandas as pd

         file_path_3 = "/Users/user/Desktop/ECO225/ECO225Project/Data/ukdetailedtimeseries2001to2
         population_df = pd.read_csv(file_path_3, header=0, low_memory=False)
```

```
In [78]: columns_to_drop = ['ladcode20', 'country', 'sex', 'population_2001', 'population_2002',
                            'population_2004', 'population_2005', 'population_2006',
                            'population_2007', 'population_2008', 'population_2009',
                            'population_2010', 'population_2011', 'population_2012',
                            'population_2013', 'Unnamed: 25']

         # Drop the specified columns
         population_df.drop(columns_to_drop, axis=1, inplace=True)
```

```
In [79]: # List of desired values
         desired_values = ['City of London', 'Barking and Dagenham', 'Barnet', 'Bexley',
                           'Brent', 'Bromley', 'Camden', 'Croydon', 'Ealing', 'Enfield',
                           'Greenwich', 'Hackney', 'Hammersmith and Fulham', 'Haringey',
                           'Harrow', 'Havering', 'Hillingdon', 'Hounslow', 'Islington',
                           'Kensington and Chelsea', 'Kingston upon Thames', 'Lambeth',
                           'Lewisham', 'Merton', 'Newham', 'Redbridge',
                           'Richmond upon Thames', 'Southwark', 'Sutton', 'Tower Hamlets',
                           'Waltham Forest', 'Wandsworth', 'Westminster']

         # Filter the DataFrame to keep only the desired values in the 'laname20' column
         population_df_filtered = population_df[population_df['laname20'].isin(desired_values)]
```

```
In [80]: melted_df = pd.melt(population_df_filtered, id_vars=['age', 'laname20'],
                             value_vars=['population_2014', 'population_2015', 'population_2016',
                                         'population_2017', 'population_2018', 'population_2019',
                                         'population_2020'],
                             var_name='year', value_name='population')

         # Convert 'year' column to integer
         melted_df['year'] = melted_df['year'].str.extract('(\d+)').astype(int)

         # Group by 'laname20' and 'year' and calculate the average age for each year
         pop_df = melted_df.groupby(['laname20', 'year']).agg({'population': 'sum'}).reset_index(
```

```
In [81]: pop_df = pd.DataFrame(pop_df)
         pop_df['laname20'] = pop_df['laname20'].str.strip().str.title()
         pop_df['laname20'] = pop_df['laname20'].replace('Westminster', 'City Of Westminster')
         final_pop = pd.merge(final_df, pop_df, left_on=['borough', 'Year_of_Transfer'], right_on
```

```
In [82]: import statsmodels.api as sm
         from statsmodels.iolib.summary2 import summary_col
         from linearmodels.iv import IV2SLS
```

```
In [83]: regression_data = final_pop.copy()
         tp_area_df = final_pop.groupby(['Area', 'Interval_2']).size().reset_index()

         tp_area_df.rename(columns={0: 'tp_area'}, inplace=True)

         regression_data= pd.merge(regression_data, tp_area_df, on=['Area', 'Interval_2'], how='l
```

```
In [84]: columns_to_remove_2 = ['Transaction_unique_identifier', 'postcode',
                                'Street', 'Locality', 'Town/City', 'District', 'laname20', 'year'
                                'PPDCategory_Type', 'County', 'Year_of_Transfer', 'Month_of_Trans
                                'Day_of_Transfer', 'Code', 'Area', 'Year']
         regression_data = regression_data.drop(columns=columns_to_remove_2)
```

In [85]:
```
regression_data['ln_price'] = np.log(regression_data['price'])
regression_data['terraced'] = (regression_data['Property_Type'] == 'T').astype(int)
regression_data['s_detached'] = (regression_data['Property_Type'] == 'S').astype(int)
regression_data['flat'] = (regression_data['Property_Type'] == 'F').astype(int)
regression_data['new'] = (regression_data['Old/New'] == 'Y').astype(int)
regression_data['leasehold'] = (regression_data['Duration'] == 'L').astype(int)
regression_data['after'] = (regression_data['Before/After'] == 'After').astype(int)
regression_data['east_london'] = regression_data['borough'].isin(['Barking And Dagenham'
                                                                   'Havering', 'Newham',
                                                                   'Redbridge', 'Tower Ha
                                                                   'Waltham Forest']).ast
regression_data.rename(columns={'Affordable Housing Supply': 'house_sup'}, inplace=True)
regression_data['terraced_x_new'] = regression_data['terraced'] * regression_data['new']
regression_data['flat_x_new'] = regression_data['flat'] * regression_data['new']
regression_data['east_x_new'] = regression_data['east_london'] * regression_data['new']
regression_data['after'] = (regression_data['Before/After'] == 'After').astype(int)
regression_data['after_x_flat_x_new'] = regression_data['flat_x_new'] * regression_data[
regression_data['population'] = regression_data['population'] / 1000
regression_data['house_sup'] = regression_data['house_sup'].str.replace(',', '').astype(
regression_data['demand_supply_ratio'] = regression_data['tp_area'] / regression_data['h
regression_data['after_x_terraced_x_new'] = regression_data['terraced_x_new'] * regressi
```

In [86]:
```
columns_to_remove_3 = ['Property_Type', 'Old/New', 'Duration', 'Interval_1', 'Interval_2
regression_data = regression_data.drop(columns=columns_to_remove_3)
regression_data['const'] = 1
rg = regression_data
```

In [87]:
```
rg['time'] = rg['Date_of_Transfer'].apply(lambda x: x.toordinal())
min_ordinal = rg['time'].min()
# Convert ordinal date data to sequential integer values starting from 1
rg['days'] = rg['time'] - min_ordinal + 1
rg.loc[:, 'days'] = rg['days'] / 100
mask = ~rg['demand_supply_ratio'].isin([np.inf, -np.inf])
columns_to_remove_4 = ['Date_of_Transfer', 'time']
rg = rg.drop(columns = columns_to_remove_4)
rg = rg[mask]
```

In [88]:
```
mean_prices = rg.groupby(['borough', 'num_parks', 'after_x_flat_x_new', 'house_sup', 'po
                          'tp_area', 'new', 's_detached', 'terraced', 'flat', 'leasehold
                          'east_london', 'terraced_x_new', 'flat_x_new', 'east_x_new', '
                          'demand_supply_ratio', 'days', 'after_x_terraced_x_new'])['ln_
mean_prices = mean_prices.reset_index()
mean_prices = mean_prices.drop(columns='borough')
```

In [89]:
```
mean_prices = mean_prices.astype(float)
```

Objective Function for OLS Model

The algorithm behind the linear regression model selects the parameters that minimize the mean squared error (MSE) function. Below is an example of the preferred specification.

$$\frac{1}{N}$$

$$\sum_{i=1}^{N}\Big(\log(\text{price}_i) - (\beta_0 + \beta_1\text{after}_t + \beta_2\text{after\_x\_flat\_x\_new}_t + \beta_3\text{after\_x\_terraced\_x\_new}_t + \beta_3\text{days}_t$$

$$+ \beta_4(\text{purchases/supplied houses})_{it} + \beta_5\text{east\_london} + \beta_6\text{east\_x\_new} + \beta_7\text{flat}_i + \beta_8\text{flat\_x\_new}_i$$

$$+ \beta_9\text{leasehold}_i + \beta_{10}\text{new}_i + \beta_{11}\text{num\_parks}_i + \beta_{12}\text{population}_{it} + \beta_{13}\text{s\_detached}_i + \beta_{14}\text{terraced}_i$$

$$+ \beta_{15}\text{terraced\_x\_new}_i)\Big)^2$$

It includes the dependent variable ln(price) as well as all independent variables (Xs). The mean squared error (MSE) function calculates the average squared difference between predicted and actual prices in the data set used for the model. A lower MSE indicates more accurate predictions, as it signifies less difference between predicted and actual property values.

Perfomance Measurements

To assess the performance of these regression models, this research uses F-statistic, R-squared and p-values.

The values of **F-statistic** above 10 usually imply that the model is overall significant and one will be able to reject the null-hypothesis.

$$H_0 : \beta_1 = \beta_1 = \beta_2 = \ldots = \beta_k = 0$$
$$H_1 : \text{not all coefficients are } 0$$

Additionally, the significance of each coefficient can be determined through its **p-value**. One, two or three stars should indicate some level of significance for each coefficient, meaning respectively that there is 5, 1 or 0.1 percent chance of having that coefficient if the actual coefficient of the model was zero. P-values help evaluate the role of each predictor in a model. Higher p-values fail to reject the null hypothesis that the coefficient is zero.

$$H_0 : \beta_k = 0$$
$$H_1 : \beta_k \neq 0$$

Finally, **R-squared** indicates the share of variance in the price that can be explained by the independent variables. Increase in R-squared implies a better fit of the model to the present data. However, adding more variables naturally increases R-squared, so this paper carefully analyzes all regressions, taking that into account.

In [90]:
```
# Create lists of variables to be used in each regression
X1 = ['const', 'terraced', 's_detached', 'flat', 'days', 'demand_supply_ratio', 'populat
X2 = ['const', 'terraced', 's_detached', 'flat', 'flat_x_new', 'terraced_x_new', 'new',
X3 = ['const', 'terraced', 's_detached', 'flat', 'leasehold',
      'days',  'demand_supply_ratio', 'population']
X4 = ['const', 'terraced', 's_detached', 'flat', 'new', 'flat_x_new', 'terraced_x_new',
      'leasehold', 'days',  'demand_supply_ratio', 'population']
# Estimate an OLS regression for each set of variables
reg1 = sm.OLS(mean_prices['ln_price'], mean_prices[X1], missing='drop').fit()
reg2 = sm.OLS(mean_prices['ln_price'], mean_prices[X2], missing='drop').fit()
reg3 = sm.OLS(mean_prices['ln_price'], mean_prices[X3], missing='drop').fit()
reg4 = sm.OLS(mean_prices['ln_price'], mean_prices[X4], missing='drop').fit()
```

Justification for variables

The dependent variable is a log of price. Taking the logarithm makes interpretation easier, especially for property purchases that vary across types, locations, and time. All other variables have linear relationships with price, so the only manipulation done is scaling for *days*. Also, instead of total purchases of property and total affordable housing supply, their ratio is used for easier interpretation and to avoid multicollinearity, as there are tight economic links between supply and demand.

- **'ln_price'**: Dependent variable
- **'const'**: Constant term

The first regression includes dummy indicators for a type of property, where a detached type is omitted because it is a reference category. They are relevant measures for the difference in price, as this research showed that some types of property tend to be more expensive than others, and linear regressions are a good estimate of relationships between dummy variables and price.

- **'terraced'**
- **'s_detached'**: for semi-detached houses
- **'flat'**

In Model 1, the estimation also considers time, population in that area during the time of purchase and a demand to supply ratio variables, which control for changes over time and market dynamics. They are useful because these variables help predict values of the purchase within each point of time and taking into account demographic and market shifts based on supply and demand.

- **'days'**: Number of days that passed since the first purchase in this data set (in March 2015). Number of days is divided by a hundred for easier interpretation.
- **'population'**: number of people living in a borough during the time that the house was purchased
- **'demand_supply_ratio'**: A number of purchased houses is divided by the number of supplied houses within the same borough and time frame. It helps to control for relative increase of demand compared to supply, which usually drives prices up based on laws of supply and demand, ceteris paribus.

In Model 2, the analysis looks into prices of different property types, holding time, population, demand to supply ratio, and age constant. It includes the dummy variable for new buildings, where the category 'old property' is omitted. It also includes two interaction variables because new flats and terraced houses exhibited different behaviors compared to old housing. These interaction variables help predict the purchase price, considering various effects on the dependent variable depending on the values of other independent variables.

- **'new'**: Indicator for new properties
- **'flat_x_new'**: Interaction term between 'flat' and 'new' because the investigation earlier has shown that flats are the most common type of new housing built and is the type that reacted most sensitively to FTB Relief.
- **'terraced_x_new'**: Interaction term between 'terraced' and 'new', as it was another type that showcased interesting price dynamics after the implementation of FTB Relief.

In Model 3, the regression predicts prices of different property types, while holding the type of tenure, market dynamics, time, and population size constant. Tenure is usually an important determinant of price because people prefer housing with more reliable property rights, such as freeholds. Higher demand usually drives prices up and creates desparities for pricces of freeholds and leaseholds.

- **'leasehold'**: Indicator for leasehold properties, the freehold type is a reference category.

In Model 4, the algorithm controls for both the age and the type of tenure, giving a more holistic image of the property market if the location is unspecified or unknown.

In [91]:
```python
from stargazer.stargazer import Stargazer
from IPython.core.display import HTML
```

OLS Models in Table 1

The first table shows estimations for the four models below. These four regression models mostly focus on features of each property, ignoring the amanities that come with location. This helps predict the average purchase price of each property type in Greater London between 2015 and 2020 based on its tenure contract and age, ignoring location within the city. Betas represent coefficients of independent variables. ϵ is an error term that captures all variables that a model does not include, therefore it partially explains variability of the data set even after controlling for certain features.

(1)

$$\widehat{ln(price)}_i = \beta_0 + \beta_1 \text{days}_t + \beta_2 (\text{purchases/supplied affordable houses})_{it} + \beta_3 \text{ flat}_i + \beta_4 \text{population}_{it} + \beta_5 \text{s\_detached}_i + \beta_6 \text{terraced}_i + \epsilon$$

(2)

$$\widehat{ln(price)}_i = \beta_0 + \beta_1 \text{days}_t + \beta_2 (\text{purchases/supplied affordable houses})_{it} + \beta_3 \text{flat}_i + \beta_4 \text{flat\_x\_new}_i + \beta_5 \text{new}_i + \beta_6 \text{population}_{it} + \beta_7 \text{s\_detached}_i + \beta_8 \text{terraced}_i + \beta_9 \text{terraced\_x\_new}_i + \epsilon$$

(3)

$$\widehat{ln(price)}_i = \beta_0 + \beta_1 \text{days}_t + \beta_2 (\text{purchases/supplied affordable houses})_{it} + \beta_3 \text{ flat}_i + \beta_4 \text{leasehold}_i + \beta_5 \text{population}_i t + \beta_6 \text{s\_detached}_i + \beta_7 \text{terraced}_i + \epsilon$$

(4)

$$\widehat{ln(price)}_i = \beta_0 + \beta_1 \text{days}_t + \beta_2 (\text{purchases/supplied affordable houses})_{it} + \beta_3 \text{flat}_i + \beta_4 \text{flat\_x\_new}_i + \beta_5 \text{leasehold}_i + \beta_6 \text{new}_i + \beta_7 \text{population}_{it} + \beta_8 \text{s\_detached}_i + \beta_9 \text{terraced}_i + \beta_{10} \text{terraced\_x\_new}_i + \epsilon$$

Table 1

In [92]:
```
stargazer_1 = Stargazer([reg1, reg2, reg3, reg4])
HTML(stargazer_1.render_html())
```

Out[92]:

				Dependent variable: ln_price
	(1)	(2)	(3)	(4)
const	14.130***	14.158***	14.129***	14.157***
	(0.010)	(0.009)	(0.010)	(0.009)
days	0.009***	0.009***	0.009***	0.009***
	(0.000)	(0.000)	(0.000)	(0.000)
demand_supply_ratio	-0.002***	-0.001***	-0.002***	-0.002***
	(0.000)	(0.000)	(0.000)	(0.000)
flat	-0.746***	-0.874***	-0.564***	-0.665***
	(0.005)	(0.006)	(0.008)	(0.008)

flat_x_new		0.362^{***}		0.364^{***}
		(0.015)		(0.015)
leasehold			-0.189^{***}	-0.220^{***}
			(0.007)	(0.007)
new		0.015		0.020
		(0.014)		(0.014)
population	-0.002^{***}	-0.002^{***}	-0.002^{***}	-0.002^{***}
	(0.000)	(0.000)	(0.000)	(0.000)
s_detached	-0.399^{***}	-0.399^{***}	-0.392^{***}	-0.391^{***}
	(0.006)	(0.006)	(0.006)	(0.006)
terraced	-0.438^{***}	-0.448^{***}	-0.419^{***}	-0.427^{***}
	(0.006)	(0.006)	(0.006)	(0.006)
terraced_x_new		0.129^{***}		0.148^{***}
		(0.018)		(0.018)
Observations	139511	139511	139511	139511
R^2	0.166	0.204	0.171	0.210
Adjusted R^2	0.166	0.204	0.171	0.210
Residual Std. Error	0.560 (df=139504)	0.548 (df=139501)	0.559 (df=139503)	0.545 (df=139500)
F Statistic	4623.798^{***} (df=6; 139504)	3967.598^{***} (df=9; 139501)	4100.818^{***} (df=7; 139503)	3713.621^{***} (df=10; 139500)
Note:				$^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$

All four models are statistically significant, as evidenced by the F-statistic, which remains consistently around 4000, therefore the null hypothesis can be rejected. The inclusion of age and type of tenure in the model leads to a slight increase in the R-squared value. A higher R-squared value indicates that the model performs better in predicting purchase values, as the independent variables explain more variance in the values.

All variable coefficients, except for "new," are statistically significant, with p-values lower than 0.01. Detached houses emerge as the most expensive option, controlling for factors such as age, time, population size, market dynamics, and tenure type. Coefficients on dummy variables for other property types suggest that old terraced, semi-detached houses, and flats are, on average, 40 to 65 percent cheaper compared to old detached houses.

The interaction variable between "new" and "flats" reveals that the premium of new-built properties is relatively higher for flats. Recently constructed flats are, on average, 40 percentage points more expensive than old flats compared to old detached houses, while holding age, time, population size, market dynamics, and tenure type constant. Terraced houses exhibit similar dynamics, but the average price difference is only 15 percentage points. Osborne (2016) argues, however, that the premium of new-builds should decrease with time due to larger supply of new housing.

The values of the ratio between total purchases and affordable housing supplied range between 1 and 10. Therefore, an additional purchase per one house supplied represents a significant change. However, the coefficient suggests that this increase is associated with only a marginal decrease of 0.02% in prices. The negative sign likely indicates that, on average, more purchases per house supplied within a borough are associated with lower price because authorities might not supply as much affordable housing when prices are low. Authorities usually provide additional housing units during a living cost crisis or recessions.

Similarly, the population coefficient is negative, indicating that an increase in population of a borough by a thousand is associated, on average, with purchases that are 0.02% cheaper. This suggests that more individuals prefer to reside where housing is more affordable.

In [93]:
```
X5 = ['const', 'terraced', 's_detached', 'flat', 'new', 'flat_x_new', 'terraced_x_new',
      'leasehold', 'days',  'demand_supply_ratio', 'population', 'east_london', 'east_x_
X6 = ['const', 'terraced', 's_detached', 'flat', 'new', 'flat_x_new', 'terraced_x_new',
      'leasehold', 'days',  'demand_supply_ratio', 'population', 'east_london', 'east_x_
X7 = ['const', 'terraced', 's_detached', 'flat', 'new', 'flat_x_new', 'terraced_x_new',
      'leasehold', 'days',  'demand_supply_ratio', 'population', 'east_london', 'after',
X8 = ['const', 'terraced', 's_detached', 'flat', 'new', 'flat_x_new', 'terraced_x_new',
      'leasehold', 'days', 'after_x_flat_x_new', 'demand_supply_ratio', 'population', 'e
      'east_x_new', 'num_parks', 'after_x_terraced_x_new']



# Estimate an OLS regression for each set of variables
reg5 = sm.OLS(mean_prices['ln_price'], mean_prices[X5], missing='drop').fit()
reg6 = sm.OLS(mean_prices['ln_price'], mean_prices[X6], missing='drop').fit()
reg7 = sm.OLS(mean_prices['ln_price'], mean_prices[X7], missing='drop').fit()
reg8 = sm.OLS(mean_prices['ln_price'], mean_prices[X8], missing='drop').fit()
```

In Model 5, the regression includes a dummy variable that indicates whether the purchase is in one of the boroughs located in East London because earlier maps showed that property tends to be more affordable there, as historically that part of the city has mostly attracted the working class. The model also has an interaction term between _east*london* and *new* because areas of East London have the highest share of new property purchases.

- **'east_london'**: an indicator for location of the purchase. West London is an omitted variable.
- **'east_x_new'**: an interaction term between East London dummy and *new*.

In Model 6, the regression additionally controls for a number of parks in the area of purchase. Harper (2019) argues that in a highly-populated and busy city, such as London, the proximity of green spaces often adds a premium to a price of a property. Few affordable housing units are located in areas with a large number of parks. Therefore, controlling for them helps to better predict the price of different housing types.

- **'num_parks'**: number of parks in a borough of a purchase

In Model 7, the analysis includes one of the main variables of interest, which controls for the FTB Relief implementation. The dummy variable after indicates whether the purchase occurred before or after that date. The indicator for purchases prior to the policy is an omitted variable. The effect of the policy is the main focus of this paper and it helps to investigate whether the FTB Relief somehow changed the market dynamics.

- **'after'**: a time period after November 27th, 2017 and March 2020.

Finally, the last model also has two interaction terms - one for *flats*, *new* and *after*, as well as a similar term for terraced houses. The key finding of the previous analysis was the immediate price spike for new flats and terraced houses after the policy implementation. Thus, this variable should help better predict the price of the purchase. However, it would mainly consider the long-term effect of the policy, rather than an immediate shock. This is the preferred model as it includes all available variables and provides the most comprehensive estimation of the FTB Relief impact on the housing market. Its estimation has the biggest R-squared, meaning that this model can explain more variability within price values than any previous model.

- **after_x_flat_x_new** : the purchase of a new flat happened after the FTB Relief was in effect
- **after_x_terraced_x_new**

OLS Models in Table 2

In the table below, I estimate the following models. These models consider additional factors that might have affected people's decisions but are not directly related to features of the building, such as green spaces near property, the location of the purchase and whether the purchase happened after the implementation of FTB Relief.

(5)

$$\begin{gathered} \widehat{ln(price)}_i = \beta_0 + \beta_1 \text{days}_t + \beta_2 (\text{purchases/supplied houses})_{it} + \beta_3 \text{east\_london} \\ + \beta_4 \text{east\_x\_new} + \beta_5 \text{flat}_i + \beta_6 \text{flat\_x\_new}_i + \beta_7 \text{leasehold}_i + \beta_8 \text{new}_i + \beta_9 \text{population}_{it} \\ + \beta_{10} \text{s\_detached}_i + \beta_{11} \text{terraced}_i + \beta_{12} \text{terraced\_x\_new}_i + \epsilon \end{gathered}$$

(6)

$$\begin{gathered} \widehat{ln(price)}_i = \beta_0 + \beta_1 \text{days}_t + \beta_2 (\text{purchases/supplied houses})_{it} + \beta_3 \text{east\_london} \\ + \beta_4 \text{east\_x\_new} + \beta_5 \text{flat}_i + \beta_6 \text{flat\_x\_new}_i + \beta_7 \text{leasehold}_i + \beta_8 \text{new}_i + \beta_9 \text{num\_parks}_i \\ + \beta_{10} \text{population}_{it} + \beta_{11} \text{s\_detached}_i + \beta_{12} \text{terraced}_i + \beta_{13} \text{terraced\_x\_new}_i + \epsilon \end{gathered}$$

(7)

$$\begin{gathered} \widehat{ln(price)}_i = \beta_0 + \beta_1 \text{after}_t + \beta_2 \text{days}_t + \beta_3 (\text{purchases/supplied houses})_{it} + \beta_4 \text{east\_london} \\ + \beta_5 \text{east\_x\_new} + \beta_6 \text{flat}_i + \beta_7 \text{flat\_x\_new}_i + \beta_8 \text{leasehold}_i + \beta_9 \text{new}_i + \beta_{10} \text{num\_parks}_i \\ + \beta_{11} \text{population}_{it} + \beta_{12} \text{s\_detached}_i + \beta_{13} \text{terraced}_i + \beta_{14} \text{terraced\_x\_new}_i + \epsilon \end{gathered}$$

(8) Preferred Specification

$$\begin{gathered} \widehat{ln(price)}_i = \beta_0 + \beta_1 \text{after}_t + \beta_2 \text{after\_x\_flat\_x\_new}_t + \beta_3 \text{after\_x\_terraced\_x\_new}_t + \beta_3 \text{days}_t \\ + \beta_4 (\text{purchases/supplied houses})_{it} + \beta_5 \text{east\_london} + \beta_6 \text{east\_x\_new} + \beta_7 \text{flat}_i \\ + \beta_8 \text{flat\_x\_new}_i + \beta_9 \text{leasehold}_i + \beta_{10} \text{new}_i + \beta_{11} \text{num\_parks}_i + \beta_{12} \text{population}_{it} \\ + \beta_{13} \text{s\_detached}_i + \beta_{14} \text{terraced}_i + \beta_{15} \text{terraced\_x\_new}_i + \epsilon \end{gathered}$$

Table 2

Note: purchases/supplied houses is represented by variable _demand_supply*ratio*

```
In [94]: stargazer_2 = Stargazer([reg5, reg6, reg7, reg8])

         HTML(stargazer_2.render_html())
```

Out[94]:

	Dependent variable: ln_price			
	(1)	(2)	(3)	(4)
after			-0.057***	-0.076***
			(0.005)	(0.005)
after_x_flat_x_new				0.113***
				(0.007)
after_x_terraced_x_new				0.098***
				(0.018)
const	14.191***	13.799***	13.785***	13.793***
	(0.009)	(0.008)	(0.008)	(0.008)
days	0.009***	0.008***	0.013***	0.013***
	(0.000)	(0.000)	(0.000)	(0.000)
demand_supply_ratio	-0.001***	-0.002***	-0.002***	-0.002***
	(0.000)	(0.000)	(0.000)	(0.000)
east_london	-0.306***	-0.072***	-0.072***	-0.072***
	(0.004)	(0.004)	(0.004)	(0.004)
east_x_new	0.039***	0.022***	0.023***	0.021***
	(0.009)	(0.008)	(0.008)	(0.008)
flat	-0.643***	-0.693***	-0.693***	-0.693***
	(0.008)	(0.007)	(0.007)	(0.007)
flat_x_new	0.359***	0.328***	0.327***	0.277***
	(0.014)	(0.012)	(0.012)	(0.013)
leasehold	-0.219***	-0.244***	-0.244***	-0.244***
	(0.006)	(0.006)	(0.006)	(0.006)
new	0.018	0.032***	0.033***	0.032***
	(0.014)	(0.012)	(0.012)	(0.012)
num_parks		0.059***	0.059***	0.059***
		(0.000)	(0.000)	(0.000)
population	-0.002***	-0.001***	-0.001***	-0.001***
	(0.000)	(0.000)	(0.000)	(0.000)
s_detached	-0.379***	-0.380***	-0.380***	-0.380***
	(0.006)	(0.005)	(0.005)	(0.005)

terraced	-0.399***	-0.445***	-0.445***	-0.445***
	(0.006)	(0.005)	(0.005)	(0.005)
terraced_x_new	0.135***	0.164***	0.164***	0.123***
	(0.017)	(0.015)	(0.015)	(0.017)
Observations	139511	139511	139511	139511
R^2	0.248	0.432	0.432	0.434
Adjusted R^2	0.248	0.432	0.432	0.434
Residual Std. Error	0.532 (df=139498)	0.463 (df=139497)	0.462 (df=139496)	0.462 (df=139494)
F Statistic	3832.950*** (df=12; 139498)	8157.717*** (df=13; 139497)	7591.919*** (df=14; 139496)	6673.796*** (df=16; 139494)
Note:				*p<0.1; **p<0.05; ***p<0.01

All models are statistically significant but F-statistic is higher after controlling for the part of the city, number of parks, and whether the purchase happened after the policy, indicating a lower chance of any coefficient being zero. The null-hypothesis can be rejected. The R-squared increases from 0.248 to around 0.43 and stays there for the last three models. This value indicates that around 43% of variance is explained by given independent variables, which is a lot considering the absence of variables that control for the size of the house, number of bedrooms and many other charachteristics.

All coefficients remain statistically significant after controlling for the amount of green spaces in the area. The coefficient of 0.059 implies that having an additional park or garden in an area of purchase is associated with an average price that is almost 6% higher, holding age, type of tenure, type of property, population, location, time and a ratio between puchases against supply fixed.

With the same variables and number of parks held constant, the property in East London, according to the three last models, is associated with an average price that is 7 percent lower for old properties and 5 percent lower for new properties compared to West London.

Lastly, the coefficients of *after*, _after_x_flat_x*new*, and _after_x_terraced_x*new* provide useful insights into the long-term effects of the FTB Relief. The respective coefficients of -0.076, 0.113 and 0.098 indicate that the average prices of purchases after the policy are associated with 7% decrease for detached and semi-detached houses, as well as old flats and old terraced houses, holding age, tenure, property type, population, part of the city, day of purchase and a demand to supply ratio constant.

The two interaction terms imply that the differences between the average prices of new flats and new terraced houses before and after the policy, compared to old detached houses, are respectively 3.7 and 2.2 percentage points lower after the policy, holding age, tenure, property type, population, part of the city, day of purchase, and demand-to-supply ratio constant.

This increase is statistically sigificant and can potentially demonstrate the long-term effect of the policy, as the predicted value of new flats and new terraced houses is slightly higher on average than before.

Machine Learning

All models in this part use the same variables as the preferred specification, which is described in the OLS Model 8.

The objective function for each region of the regression tree:

$$\min_{j,s}\left[\sum_{i:x_{i,j}\leq s, x_i\in R1}(ln(price)_i - ln(\hat{price})_{R1})^2 + \sum_{i:x_{i,j}>s, x_i\in R2}(ln(price)_i - ln(\hat{price})_{R2})^2\right]$$

A rectangular region R is defined initially, encompassing all values of X. To create a branch of a tree represented by a smaller rectangular region R_n encompassing a share of values, a feature and location are then selected for splitting, with the aim of minimizing mean squared error (MSE). This process is repeated iteratively to generate the number of branches that were specified by the depth of the tree. Then, the tree is pruned, which means selectively removing branches or nodes that do not improve the model's performance. The main purpose of pruning is to avoid overfitting, which usually happens when the model predicts the existing data well but performs badly on a new set of data.

$$\min_{tree\subset T}\sum(\hat{f}(x) - ln(price)^2 + \alpha|\text{terminal} \quad \text{nodes in tree}|$$

- **Maximum tree depth** contols the number of levels starting with the root node to the leaf nodes, which determines the complexity of the tree.
- **Minimum leaf size** decides the required minimum number of samples in each node, influencing how fine or general the region of a new rectengular is after the tree's partitioning.
- α is the regularization parameter. It sometimes balances the trade-off between model complexity and accuracy. Alpha penalizes the complexity of the tree.

Higher maximum depth of the tree makes the model more complex and reduced MSE but it is hard to visualise the model with more levels. Increase in maximum depth also risks overfitting.

Increasing the minimum leaf size motivates more generalized rectengular regions of the data to reduce chances of overfitting. Setting a larger minimum leaf size might be challenging if a total sample size is small. Also, this approach risks underfitting, as the model would ignore some finer but still valid data trends.

Higher α penalizes the complexity of the tree more agressively, resulting in simpler and more general models. However, if it is too high, the MSE can be relatively large, indicating large error of prediction.

The minimum leaf size for the model below is 300, which means that each leaf node in the decision tree must contain at least 300 samples (data points). The choice of 300 is feasible, as the total sample size is 20 thousand data points. A maximum tree depth is 3 for easier visualisation, whereas α is 0.00, meaning that the algorithm applies no penalty for the complexity of the tree to minimize MSE.

Regression Tree and Importance Matrix

All decision models will include independent variables from the regression model 8 because all coefficients of those variables are statistically significant. Inluding those Xs leads to the highest R-squared and generates useful insights into the long-term effects of the FTB Relief policy in the UK after the implementation.

In [118…
```
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn import (
    linear_model, metrics, pipeline, model_selection
)
from sklearn.metrics import mean_squared_error
```

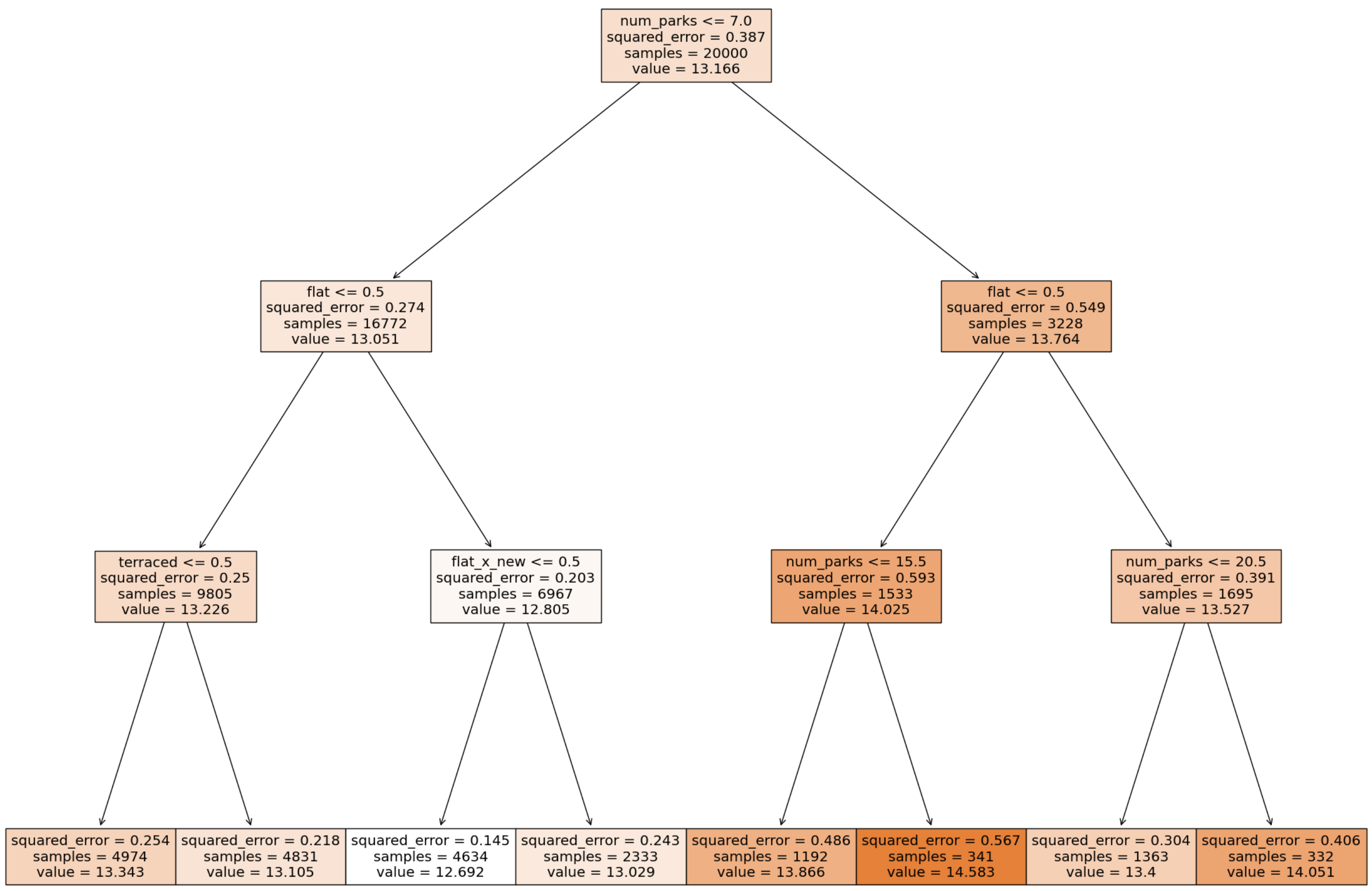
In [101…
```
subset_rg = mean_prices.sample(n=20000, random_state=42)
```

In [109…
```
X = subset_rg.drop(['ln_price', 'leasehold', "house_sup", 'tp_area'], axis=1).copy()
for col in list(X):
    X[col] = X[col].astype(float)
y = subset_rg['ln_price']
```

In [122…
```
from sklearn.linear_model import LinearRegression
lr_model = LinearRegression()
lr_model.fit(X, y)
y_pred_linear = lr_model.predict(X)
full_mse = metrics.mean_squared_error(y, y_pred_linear)
```

In [124…
```
from sklearn import tree
house_tree = tree.DecisionTreeRegressor(max_depth=3, min_samples_leaf=300, ccp_alpha=0.0
y_pred_tree = house_tree.predict(X)
mse_decision = metrics.mean_squared_error(y, y_pred_tree)
```

In [125…
```
feature_names = X.columns.tolist()
plt.figure(figsize=(25,20))
tree.plot_tree(house_tree, feature_names=feature_names, filled=True)
plt.show()
```



The regression decision tree above has 3 levels, where each split minimizes the MSE.

The first feature of the split is the number of parks, meaning that it is the most important feature for the model predictions. The value is 7 because, according to one of the previous graphs, most boroughs have fewer than 9 parks, and the split divides the sample into sixteen and four thousand data points. The model predicts that the property with more than 7 parks in a borough is significantly more expensive. This corresponds to the previous findings, where boroughs with more than 10 parks had one of the most expensive average prices for property in London.

The next highly important feature for both splits was whether the property type is a flat, as they are predicted to be the cheapest housing type. The threshold is 0.5 because it is a dummy variable and possible values are only 0 and 1.

Then, if a property is a flat and the number of parks around is less than 7, the next most important variable is the interaction term between flat and new. It also has a threshold of 0.5 because it is a dummy variable, where potential answers are 'a new flat' or 'not a new flat', represented by 1 and 0.

If the flat is new, the average predicted value is the lowest among all 8 subgroups and has also the lowest squared error, indicating the best fit of the model among all 8 value subgroups. The highest value and error are for the subgroup of non-flats with more than 16 parks in the area. The possible reason is the size of the sample with only 300 values, but it can also indicate higher variance for property prices in the high-end segment of the market, where difference sometimes can be millions of pounds.

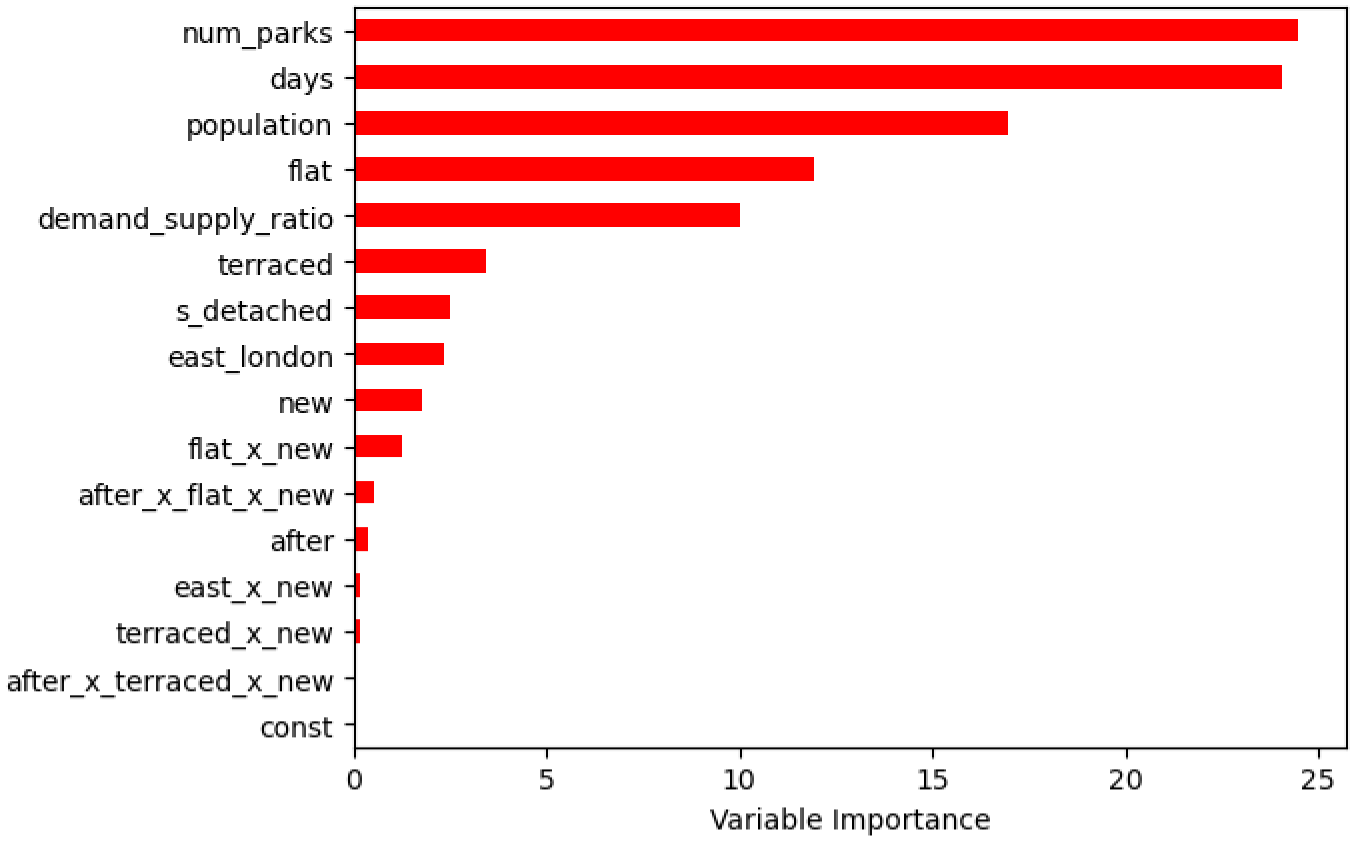
The overall squared error between 0.2 and 0.56 for the dependent variable *ln(price)*, which translates into difference of a couple thousand dollars squared, implies that the model predicts actual values quite well but it is not too precise to indicate overfitting.

RandomForest

In [126…
```
regr3 = RandomForestRegressor(max_features=5, random_state=42)
regr3.fit(X, y)
pred = regr3.predict(X)
random_mse = mean_squared_error(y, pred)
```

The Importance Matrix

In [127…
```
Importance = pd.DataFrame({'Importance':regr3.feature_importances_*100}, index=X.columns
Importance.sort_values('Importance', axis=0, ascending=True).plot(kind='barh', color='r'
plt.xlabel('Variable Importance')
plt.gca().legend_ = None
```



Overall, mostly green spaces, population density, and time of purchase determine how much buyers pay. All other variables had significantly smaller importance share. Whether a property is a flat or not is a more significant predictor of the outcome variable in the model than any other dummy variable for the property type. _Flat_x*new* and _after_x_flat_x*new* are the most important features among all interaction variables, signifying weak but present relationships.

Error of Prediction

In [128…
```
errors = {'Model': ['Linear', 'Regression Tree', 'Random Forest'],
          'Mean Squared Error': [full_mse, mse_decision, random_mse]}
df_errors = pd.DataFrame(errors)
```

```
df_errors['Mean Squared Error'] = df_errors['Mean Squared Error'].round(3)
df_errors
```

Out[128]:

	Model	Mean Squared Error
0	Linear	0.219
1	Regression Tree	0.244
2	Random Forest	0.025

Comparing the results from running a regression with those from running a regression tree highlights the strengths and weaknesses of each approach. The differencce in MSE for each model indicates that OLS model is slightly better in predicting than the regression tree but still not as good as Random Forest Model.

OLS regression is a powerful tool for estimating the linear relationship between the price of property and independent variables because it provides coefficients that represent the marginal effect of a change in one variable while holding others constant. All these coefficients are statistically significant and offer economic interpretation, enabling hypothesis testing and informing policy decisions. For instance, while aiming to estimate the impact of the FTB Relief, this paper examines variables such as *after*, _after_x_flat_x*new*, and _after_x_terraced_x*new*. Even though they are not the most important variables, according to the matrix graph, they provide the answer to the research question of this paper. The analysis reveals a significant noteworthy decrease in price percentage differences compared to detached houses, ranging from 2 to 4 percentage points, for newly constructed terraced houses and flats after the policy implementation. The change is economically significant as it translates into a couple thousands of pounds.

In contrast, regression trees offer a different perspective by capturing complex and heterogeneous relationships in the data. They display decision-making processes, showcasing the importance of each variable in predicting house prices. Unlike OLS regression, where coefficients are estimated for each independent variable, regression trees identify key features and their thresholds during partitioning, focusing on accuracy of prediction over estimation of parameters.

For instance, while the OLS model identifies all coefficients as highly statistically significant, it emphasizes the importance of property types in price prediction, with relatively smaller effects for location and time variables. On the other hand, the regression tree prioritizes variables differently, with only a dummy for flats being significant, while the presence of parks and the day of purchase play a crucial role in determining prices.

Besides, the highly important variables in the regression trees, such as day of purchase, population and number of parks, had the standard error of 0 in OLS Model, meaning that their estimates perfectly predicted the true population parameter every time.

Considering the research question, OLS enabled the analysis to focus on the key variables of interest, whereas the regression tree better displayed the overall picture to help understand the unique market dynamics in Greater London.

Conclusion and Further Work

The introduction of the First Time Buyers' Relief policy on November 22, 2017, had a significant impact only on housing prices of two property types in the Greater London Area. There was a significant short-term price surge in newly-built flats and terraced houses in the quarter including November 2017, which did not follow the usual trend of price decrease during cold months. These types of property and the price increase correlate with the timing of the policy introduction and tax relief eligibility criteria. The findings of this paper support an economic hypothesis that the introduction of the First Time Buyers' Relief policy in November 2017 significantly impacted affordable housing options in the Greater London Area, particularly affecting newly-built flats and terraced houses.

First-time buyers, usually younger individuals with fewer savings, searched for more affordable property options, potentially contributing to increased demand and subsequent price increases of new flats and new terraced houses. Also, even though the effect on new flats and terraced houses is present, the initial divergence from usual trends lasted only for a few months.

The primary constraint of this study lies in the absence of comprehensive official data detailing gentrification processes across each borough. Distinguishing between the price surges attributed to the policy's impact and those stemming from inherent gentrification poses a considerable challenge. Gentrification, occurring organically over time, tends to elevate average property prices, complicating the differentiation process, particularly when considering the broader market dynamics influencing the cityscape (Guardian, 2016).

While the FTB Relief policy stimulated demand for certain property types, particularly among first-time buyers, it did not affect all London districts equally. The composition of the housing market in East and West London is very diverse, and certain types of property are historically more concentrated in some areas; for example, central parts of London have more flats, and West London is more attractive for consumers willing to pay higher prices. East London usually offers more affordable housing, as historically it used to be mainly a working-class area. The higher increase in property prices in East London can suggest higher demand for more affordable housing, which corresponds with the hypothesis of the policy, that the tax relief should lead to higher demand.

The analysis suggested that property markets with a 'park premium' did not experience significant changes despite FTB Relief implementation, while boroughs with fewer parks experienced greater price fluctuations. This can be explained by the policy target of the lower-priced property market. Also, Tower Hamlets, Newham, and Southwark are significant hubs of affordable housing provision post-FTB Relief, with Tower Hamlets potentially showing minor price adjustments compared to Newham, which has seen an extreme rise in property prices. It might indicate the relevance of higher housing supply elasticity, as more affordable housing is built annually in Tower Hamlets than in Newham. There was also a surge in affordable housing supply post-FTB Relief, which might reflect market responsiveness to increased demand, highlighting the connection between supply and demand in the housing market.

The OLS models and regression trees suggest that the implementation of the policy had a significant long-term effect. After the FTB Relief, the difference between average prices of new terraced houses and flats compared to old detached houses in Western London, which is the reference group, decreased by 2 to 4 percentage points, taking the type of tenure, the number of parks, population, and market dynamics into account. Osborne (2016) argued that higher supply of the new housing should decrease the premium of new-builds but policy implementation increased demand, which likely counteracted the recent construction.

Finally, the FTB Relief removed the 2% and 5% property tax from purchases below £300,000 and kept the 5% tax for housing valued between £300,000 and up to £500,000. The long-term effect determined by the regression model indicates that first-time buyers did not significantly benefit from the tax break, as it was offset by the increase in the average price of affordable housing options due to upward shifts of demand.

Some further research might conduct Difference-in-Differences modeling by comparing the sample that was affected by the policy to the control group (Shopov, 2023). Researchers can also investigate the number of purchases between March 2015 and March 2020 in Greater London rather than the market price of the property, as it can generate more insights on the number of people that acquired a home and benefited from the scheme. Economists might consider the effect of tax relief on other regions of England and compare results with Greater London to evaluate the spatial efficiency of the policy (McKee et al. 2016). Including the data on gentrification into multiple regression model and decision trees will also improve the reliability of findings. This evaluation might help policymakers determine what measures to implement to increase housing affordability and social mobility.

References

1. Bolster, A. (2011). Evaluating the Impact of Stamp Duty Land Tax First Time Buyer's Relief. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dbc6dd9a2a67fcd3 e76bd3ed699d3533de147d24
2. Bone, J., & O'Reilly, K. (2010). No place called home: the causes and social consequences of the UK housing 'bubble.' The British Journal of Sociology, 61(2), 231–255. https://doi.org/10.1111/j.1468-4446.2010.01311.x
3. Bryant, G. E. (2012). Regional first-time buyers of housing: what is stopping us? [Review of Regional first-time buyers of housing: what is stopping us?].
4. Greater London Authority. (2011). Ward Profiles and Atlas – London Datastore. London.gov.uk. https://data.london.gov.uk/dataset/ward-profiles-and-atlas
5. Guardian readers, & Perry, F. (2016, January 13). "You are left with no choice but to leave" – your stories of long-term gentrification. The Guardian; The Guardian. https://www.theguardian.com/cities/2016/jan/13/no-choice-leave-displacement-tales-long-term-gentrification-six-cities
6. Gulliver, B. (2022, May 29). Area named best place to live in London as council tax goes furthest. My London. https://www.mylondon.news/news/cost-of-living/london-property-area-named-best-24070437
7. Harper, P. (2019). Parks premium: how does green space affect London house prices? - Which? News. Which? https://www.which.co.uk/news/article/parks-premium-how-does-green-space-affect-london-house-prices-a2pFT8b8O9yX
8. HM Land Registry Open Data. (2023). Landregistry.data.gov.uk. https://landregistry.data.gov.uk/
9. Lawrence, I. (2022, December 5). This new map shows London's most deprived areas. Time out London. https://www.timeout.com/london/news/this-new-map-shows-londons-most-deprived-areas-120522
10. Manton, C. (2023, March 6). The Most Expensive Boroughs in London. Best Gapp. https://bestgapp.co.uk/news/most-expensive-area-in-london/
11. Masey, A. (2019, January 16). This well-connected hub boasts one of London's most scenic stretches. Evening Standard. https://www.standard.co.uk/homesandproperty/where-to-live/living-

in-hammersmith-area-guide-to-homes-schools-and-transport-a126821.html

12. McKee, K., Muir, J., & Moore, T. (2016). Housing policy in the UK: the importance of spatial nuance. Housing Studies, 32(1), 60–72. https://doi.org/10.1080/02673037.2016.1181722
13. Ngai, L. R., & Tenreyro, S. (2014). Hot and Cold Seasons in the Housing Market. American Economic Review, 104(12), 3991–4026. https://doi.org/10.1257/aer.104.12.3991
14. Osborne, H. (2016, March 7). Glut of new-build properties leads to falling premiums in central London. The Guardian. https://www.theguardian.com/money/2016/mar/07/glut-new-build-properties-falling-premiums-central-london-oversupply
15. Registered Parks & Gardens | Historic England. (n.d.). Historicengland.org.uk. https://historicengland.org.uk/listing/what-is-designation/registered-parks-and-gardens/
16. Sabater, A., & Finney, N. (2022). Age segregation and housing unaffordability: Generational divides in housing opportunities and spatial polarisation in England and Wales. Urban Studies. https://doi.org/10.1177/00420980221121088
17. Shopov, R., Howell, H., & Claridge, F. (2023, May 23). Evaluating the impact of the 2017 Stamp Duty Land Tax First Time Buyers' Relief [Review of Evaluating the impact of the 2017 Stamp Duty Land Tax First Time Buyers' Relief]. Gov.uk. https://www.gov.uk/government/publications/evaluating-the-impact-of-the-2017-stamp-duty-land-tax-first-time-buyers-relief/evaluating-the-impact-of-the-2017-stamp-duty-land-tax-first-time-buyers-relief#:~:text=The%202011%20relief%20exempted%20FTB,rev iew%20its%20existing%20tax%20reliefs.
18. Szumilo, N. (2018). The spatial consequences of the housing affordability crisis in England. Environment and Planning A: Economy and Space, 51(6), 1264–1286. https://doi.org/10.1177/0308518x18811671
19. Wilson , W. (2022). Rent setting: Social housing (England) [Review of Rent setting: Social housing (England)]. House of Commons, Library. commonslibrary.parliament.uk