In [68]:
```python
import pandas as pd
import seaborn as sns
```

In [13]:
```python
df = pd.read_csv(r"F:\Maven Python Project\Airbnb Data\Listings.csv", encoding="ISO-8859-1", low_memory=False)
```

In [16]:
```python
df
```

Out[16]:

	listing_id	name	host_id	host_since	host_location	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_total_listings_count	...	minimum_nights
0	281420	Beautiful Flat in le Village Montmartre, Paris	1466919	2011-12-03	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
1	3705183	39 mÂ² Paris (Sacre CÅ“ur)	10328771	2013-11-29	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
2	4082273	Lovely apartment with Terrace, 60m2	19252768	2014-07-31	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
3	4797344	Cosy studio (close to Eiffel tower)	10668311	2013-12-17	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
4	4823489	Close to Eiffel Tower - Beautiful flat : 2 rooms	24837558	2014-12-14	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
...	...	...	...	...	...	...	...	...	...	...	...	...
279707	38338635	Appartement T2 neuf prÃ¨s du tram T3a Porte ...	31161181	2015-04-13	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	1
279708	38538692	Cozy Studio in Montmartre	10294858	2013-11-27	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	7
279709	38683356	Nice and cosy mini-appartement in Paris	2238502	2012-04-27	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	6
279710	39659000	Charming apartment near Rue Saint Maur / Oberk...	38633695	2015-07-16	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	3
279711	40219504	Cosy apartment with view on Canal St Martin	6955618	2013-06-17	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2

279712 rows × 33 columns

In [17]:
```python
df.head()
```

Out[17]:

	listing_id	name	host_id	host_since	host_location	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_total_listings_count	...	minimum_nights
0	281420	Beautiful Flat in le Village Montmartre, Paris	1466919	2011-12-03	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
1	3705183	39 mÂ² Paris (Sacre CÅ“ur)	10328771	2013-11-29	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
2	4082273	Lovely apartment with Terrace, 60m2	19252768	2014-07-31	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
3	4797344	Cosy studio (close to Eiffel tower)	10668311	2013-12-17	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
4	4823489	Close to Eiffel Tower - Beautiful flat : 2 rooms	24837558	2014-12-14	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2

5 rows × 33 columns

Profile & QA the data

In [18]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 279712 entries, 0 to 279711
Data columns (total 33 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   listing_id                   279712 non-null  int64
 1   name                         279539 non-null  object
 2   host_id                      279712 non-null  int64
 3   host_since                   279547 non-null  object
 4   host_location                278872 non-null  object
 5   host_response_time           150930 non-null  object
 6   host_response_rate           150930 non-null  float64
 7   host_acceptance_rate         166625 non-null  float64
 8   host_is_superhost            279547 non-null  object
 9   host_total_listings_count    279547 non-null  float64
 10  host_has_profile_pic         279547 non-null  object
 11  host_identity_verified       279547 non-null  object
 12  neighbourhood                279712 non-null  object
 13  district                     37012 non-null   object
 14  city                         279712 non-null  object
 15  latitude                     279712 non-null  float64
 16  longitude                    279712 non-null  float64
 17  property_type                279712 non-null  object
 18  room_type                    279712 non-null  object
 19  accommodates                 279712 non-null  int64
 20  bedrooms                     250277 non-null  float64
 21  amenities                    279712 non-null  object
 22  price                        279712 non-null  int64
 23  minimum_nights               279712 non-null  int64
 24  maximum_nights               279712 non-null  int64
 25  review_scores_rating         188307 non-null  float64
 26  review_scores_accuracy       187999 non-null  float64
 27  review_scores_cleanliness    188047 non-null  float64
 28  review_scores_checkin        187941 non-null  float64
 29  review_scores_communication  188025 non-null  float64
 30  review_scores_location       187937 non-null  float64
 31  review_scores_value          187927 non-null  float64
 32  instant_bookable             279712 non-null  object
dtypes: float64(13), int64(6), object(14)
memory usage: 70.4+ MB
```

In [22]:
```python
#here the only DateTime col is host_since but its datatype is "int", we have to change it to datatype "DateTime"

df["host_since"] = pd.to_datetime(df["host_since"])

df.info() #as we can see host_simnce is now changed to datatype "datetime".
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 279712 entries, 0 to 279711
Data columns (total 33 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   listing_id                   279712 non-null  int64
 1   name                         279539 non-null  object
 2   host_id                      279712 non-null  int64
 3   host_since                   279547 non-null  datetime64[ns]
 4   host_location                278872 non-null  object
 5   host_response_time           150930 non-null  object
 6   host_response_rate           150930 non-null  float64
 7   host_acceptance_rate         166625 non-null  float64
 8   host_is_superhost            279547 non-null  object
 9   host_total_listings_count    279547 non-null  float64
 10  host_has_profile_pic         279547 non-null  object
 11  host_identity_verified       279547 non-null  object
 12  neighbourhood                279712 non-null  object
 13  district                     37012 non-null   object
 14  city                         279712 non-null  object
 15  latitude                     279712 non-null  float64
 16  longitude                    279712 non-null  float64
 17  property_type                279712 non-null  object
 18  room_type                    279712 non-null  object
 19  accommodates                 279712 non-null  int64
 20  bedrooms                     250277 non-null  float64
 21  amenities                    279712 non-null  object
 22  price                        279712 non-null  int64
 23  minimum_nights               279712 non-null  int64
 24  maximum_nights               279712 non-null  int64
 25  review_scores_rating         188307 non-null  float64
 26  review_scores_accuracy       187999 non-null  float64
 27  review_scores_cleanliness    188047 non-null  float64
 28  review_scores_checkin        187941 non-null  float64
 29  review_scores_communication  188025 non-null  float64
 30  review_scores_location       187937 non-null  float64
 31  review_scores_value          187927 non-null  float64
 32  instant_bookable             279712 non-null  object
dtypes: datetime64[ns](1), float64(13), int64(6), object(13)
memory usage: 70.4+ MB
```

In [23]:
```python
df
```

Out[23]:

	listing_id	name	host_id	host_since	host_location	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_total_listings_count	...	minimum_nights
0	281420	Beautiful Flat in le Village Montmartre, Paris	1466919	2011-12-03	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
1	3705183	39 mÂ² Paris (Sacre CÅ“ur)	10328771	2013-11-29	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
2	4082273	Lovely apartment with Terrace, 60m2	19252768	2014-07-31	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
3	4797344	Cosy studio (close to Eiffel tower)	10668311	2013-12-17	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
4	4823489	Close to Eiffel Tower - Beautiful flat : 2 rooms	24837558	2014-12-14	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2
...	...	...	...	...	...	...	...	...	...	...	...	...
279707	38338635	Appartement T2 neuf prÃ¨s du tram T3a Porte ...	31161181	2015-04-13	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	1
279708	38538692	Cozy Studio in Montmartre	10294858	2013-11-27	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	7
279709	38683356	Nice and cosy mini-appartement in Paris	2238502	2012-04-27	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	6
279710	39659000	Charming apartment near Rue Saint Maur / Oberk...	38633695	2015-07-16	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	3
279711	40219504	Cosy apartment with view on Canal St Martin	6955618	2013-06-17	Paris, Ile-de-France, France	NaN	NaN	NaN	f	1.0	...	2

279712 rows × 33 columns

In [30]:
```python
# Filter for rows where 'City' is 'Paris'

paris_df = df[df["city"] == 'Paris'][['host_since', 'neighbourhood', 'city', 'accommodates', 'price']]
paris_df
```

Out[30]:

	host_since	neighbourhood	city	accommodates	price
0	2011-12-03	Buttes-Montmartre	Paris	2	53
1	2013-11-29	Buttes-Montmartre	Paris	2	120
2	2014-07-31	Elysee	Paris	2	89
3	2013-12-17	Vaugirard	Paris	2	58
4	2014-12-14	Passy	Paris	2	60
...	...	...	...	...	...
279707	2015-04-13	Observatoire	Paris	2	120
279708	2013-11-27	Buttes-Montmartre	Paris	2	60
279709	2012-04-27	Buttes-Montmartre	Paris	2	50
279710	2015-07-16	Popincourt	Paris	2	105
279711	2013-06-17	Enclos-St-Laurent	Paris	2	70

64690 rows × 5 columns

In [32]:
```python
paris_df.info()

#here the total records each row should be 64690 but col "host_since" is only with 64657 entris, that means there are missing values in this col.
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 64690 entries, 0 to 279711
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   host_since     64657 non-null  datetime64[ns]
 1   neighbourhood  64690 non-null  object
 2   city           64690 non-null  object
 3   accommodates   64690 non-null  int64
 4   price          64690 non-null  int64
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 3.0+ MB
```

In [36]:
```python
paris_df.isna().sum()

#in col host_since there are 33 NA values.
```

Out[36]:
```
host_since       33
neighbourhood     0
city              0
accommodates      0
price             0
dtype: int64
```

In [37]:
```python
paris_df.describe()
```

Out[37]:

	accommodates	price
count	64690.000000	64690.000000
mean	3.037997	113.096445
std	1.588766	214.433668
min	0.000000	0.000000
25%	2.000000	59.000000
50%	2.000000	80.000000
75%	4.000000	120.000000
max	16.000000	12000.000000

In [39]:
```python
paris_df.query("accommodates == 0").count() #count is for counting.

#The .query() method in pandas is a powerful tool for filtering DataFrame rows based on a boolean expression.
#It allows you to write a concise and readable string expression that defines the filtering criteria.
```

Out[39]:
```
host_since       54
neighbourhood    54
city             54
accommodates     54
price            54
dtype: int64
```

In [40]:
```python
paris_df.query("price == 0").count()
```

Out[40]:
```
host_since       62
neighbourhood    62
city             62
accommodates     62
price            62
dtype: int64
```

Prepare the data for visualization

In [43]:
```python
#Create a table named paris_listings_neighbourhood that groups Paris listings by 'neighbourhood' and
#calculates the mean price (sorted low to high)


paris_df_neighbourhood = (
    paris_df
    .groupby("neighbourhood") #grouping the paris data by neighbourhood.
    .agg({"price" : "mean"}) #aggregating(mean) on basis of price for each neighbourhood.
    .sort_values("price") #asc sorting the price on basis of neighbourhood [cheapest to expensive]

)

paris_df_neighbourhood
```

Out[43]:

	price
neighbourhood	
Menilmontant	74.942257
Buttes-Chaumont	82.690182
Buttes-Montmartre	87.209479
Reuilly	89.058402
Popincourt	90.559459
Gobelins	98.110184
Observatoire	101.866801
Batignolles-Monceau	102.612702
Enclos-St-Laurent	102.967156
Vaugirard	106.831330
Opera	119.038644
Pantheon	122.662150
Temple	138.446823
Hotel-de-Ville	144.472110
Bourse	149.496801
Luxembourg	155.638639
Palais-Bourbon	156.856578
Passy	161.144635
Louvre	175.379972
Elysee	210.536765

In [57]:
```python
#Create a table named paris_listings_accomodations, filter down to the most expensive neighbourhood,
#group by the ‘accommodations’ column, and add the mean price for each value of ‘accommodates’ (sorted low to high)


paris_df_accommodates =(
    paris_df
    .query("neighbourhood == 'Elysee'") #working on basis of most expensive neighbourhood i.e. Elysee
    .groupby("accommodates") #group by on basis of accommodates on Elysee's data
    .agg({"price": "mean"}) #price aggragated(mean) on basis of price on Elysee's data
    .sort_values("price") #sorted asc on price on basis of on Elysee's data

)

paris_df_accommodates
```

Out[57]:

	price
accommodates	
0	0.000000
1	79.522222
3	152.828767
2	155.103352
4	212.096070
5	328.817073
6	355.508571
8	405.518519
7	411.538462
9	440.272727
10	500.857143
12	529.625000
16	800.000000
11	805.000000
13	842.500000
14	971.000000

In [63]:
```python
#Create a table called paris_listings_over_time grouped by the ‘host_since’ year,
#and calculate the average price and count of rows representing the number of new hosts


paris_df_over_time = (
    paris_df
    .set_index("host_since") #setting "host_since" as index.
    .resample("Y")#In pandas, the resample method is a powerful tool for working with time series data.here we are doing it for Y:Year.
    .agg({
        "neighbourhood": "count",
        "price": "mean"
    })

)

paris_df_over_time
```

Out[63]:

	neighbourhood	price
host_since		
2008-12-31	4	77.750000
2009-12-31	106	159.641509
2010-12-31	416	125.031250
2011-12-31	1339	124.828230
2012-12-31	4592	111.578615
2013-12-31	8142	107.096414
2014-12-31	10922	100.253800
2015-12-31	12147	103.646250
2016-12-31	8871	114.159847
2017-12-31	4585	108.658888
2018-12-31	4294	138.209362
2019-12-31	5694	129.757113
2020-12-31	3412	141.456038
2021-12-31	133	93.488722

In [58]:
```python
paris_df
```

Out[58]:

	host_since	neighbourhood	city	accommodates	price
0	2011-12-03	Buttes-Montmartre	Paris	2	53
1	2013-11-29	Buttes-Montmartre	Paris	2	120
2	2014-07-31	Elysee	Paris	2	89
3	2013-12-17	Vaugirard	Paris	2	58
4	2014-12-14	Passy	Paris	2	60
...	...	...	...	...	...
279707	2015-04-13	Observatoire	Paris	2	120
279708	2013-11-27	Buttes-Montmartre	Paris	2	60
279709	2012-04-27	Buttes-Montmartre	Paris	2	50
279710	2015-07-16	Popincourt	Paris	2	105
279711	2013-06-17	Enclos-St-Laurent	Paris	2	70

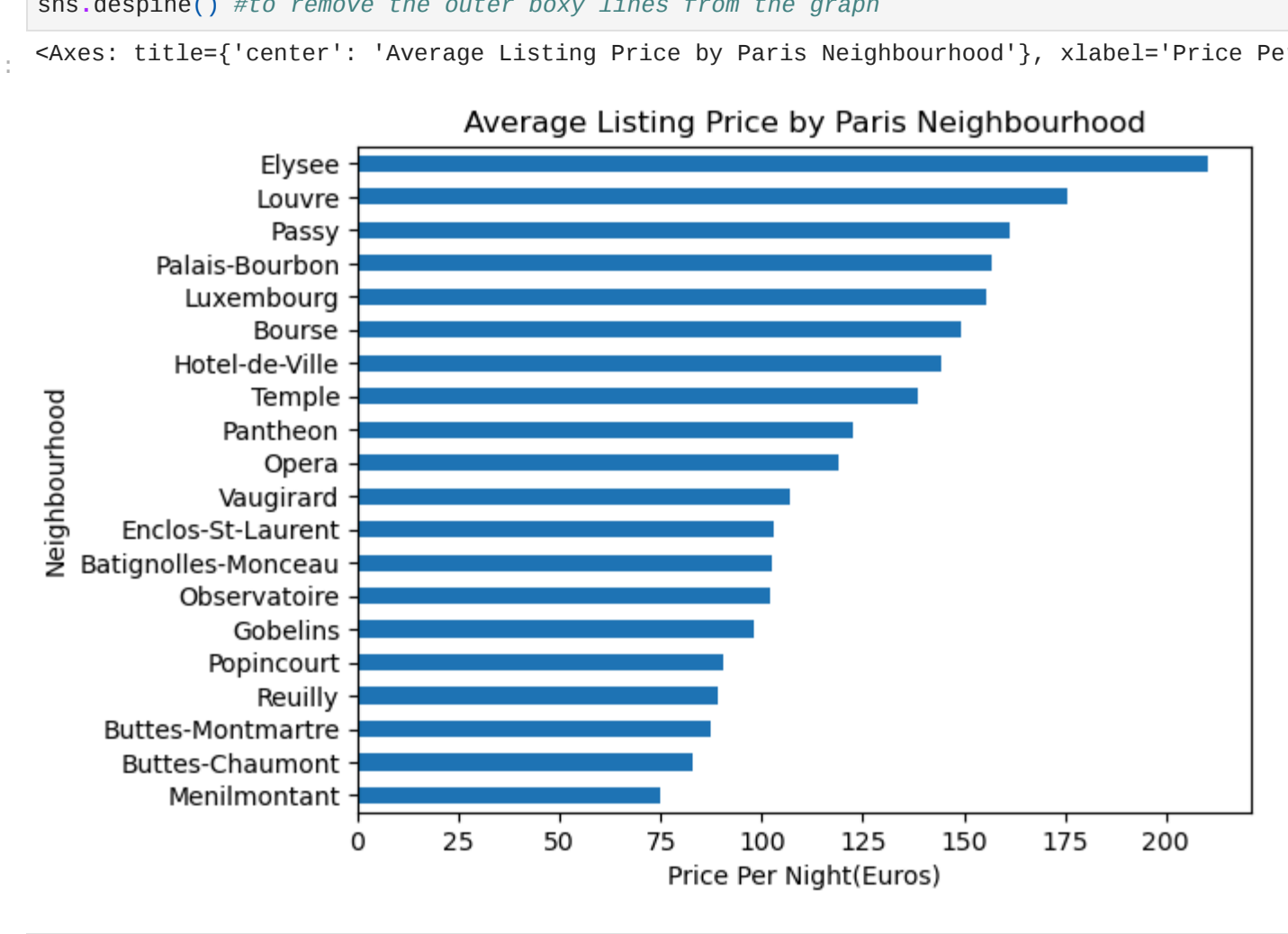
64690 rows × 5 columns

In [ ]:

Visualize the data and summarize findings

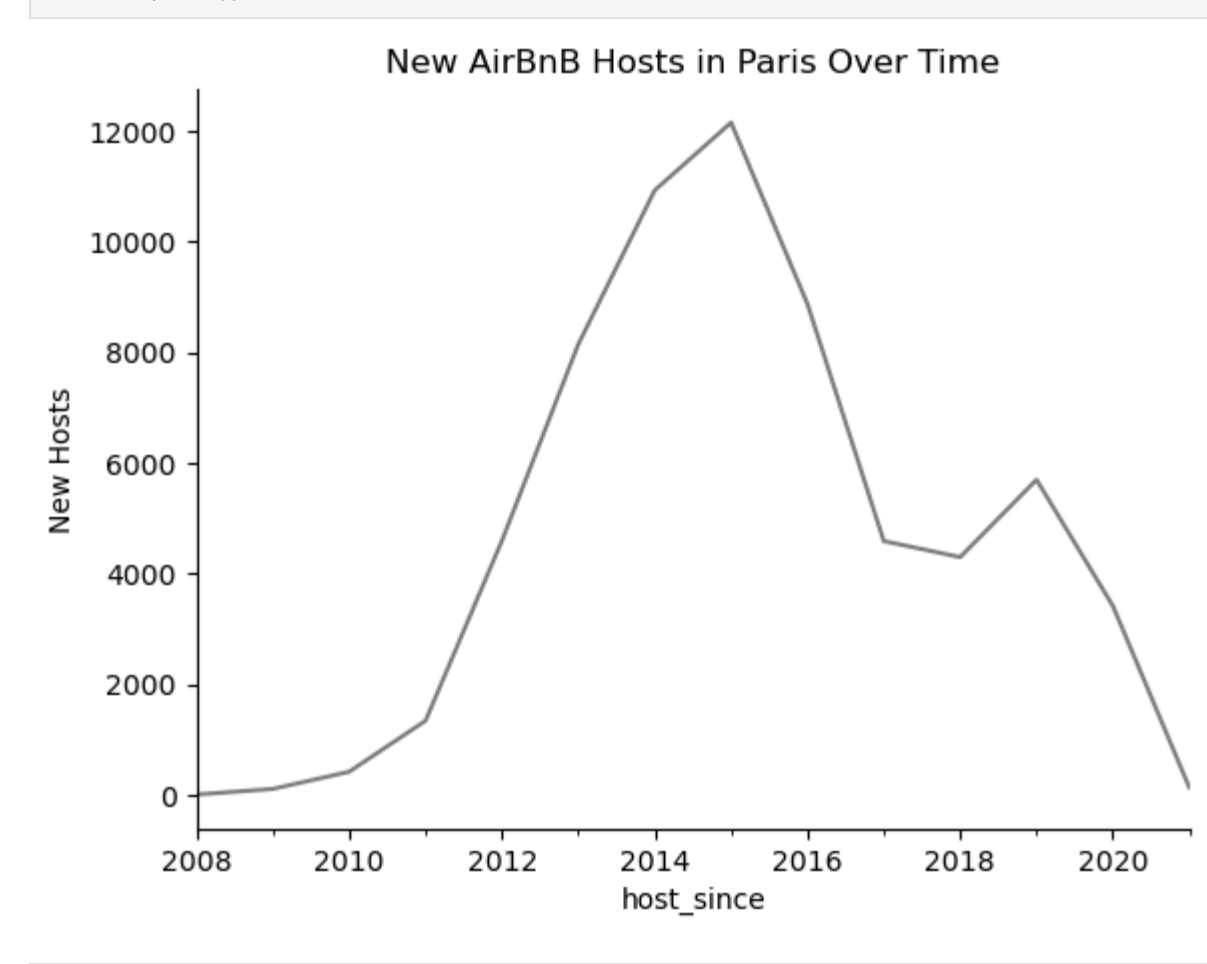
In [81]:
```python
#Create a horizontal bar chart of the average price by neighborhood in Paris,
#and make sure to add a title and change axis labels as needed

paris_df_neighbourhood.plot.barh(
    title = "Average Listing Price by Paris Neighbourhood",
    xlabel = "Price Per Night(Euros)",
    ylabel = "Neighbourhood",
    legend = None #the legend was "Price" and it is obvious that  X  axis it is the Price.
)

sns.despine() #to remove the outer boxy lines from the graph
```

Out[81]: `<Axes: title={'center': 'Average Listing Price by Paris Neighbourhood'}, xlabel='Price Per Night(Euros)', ylabel='Neighbourhood'>`

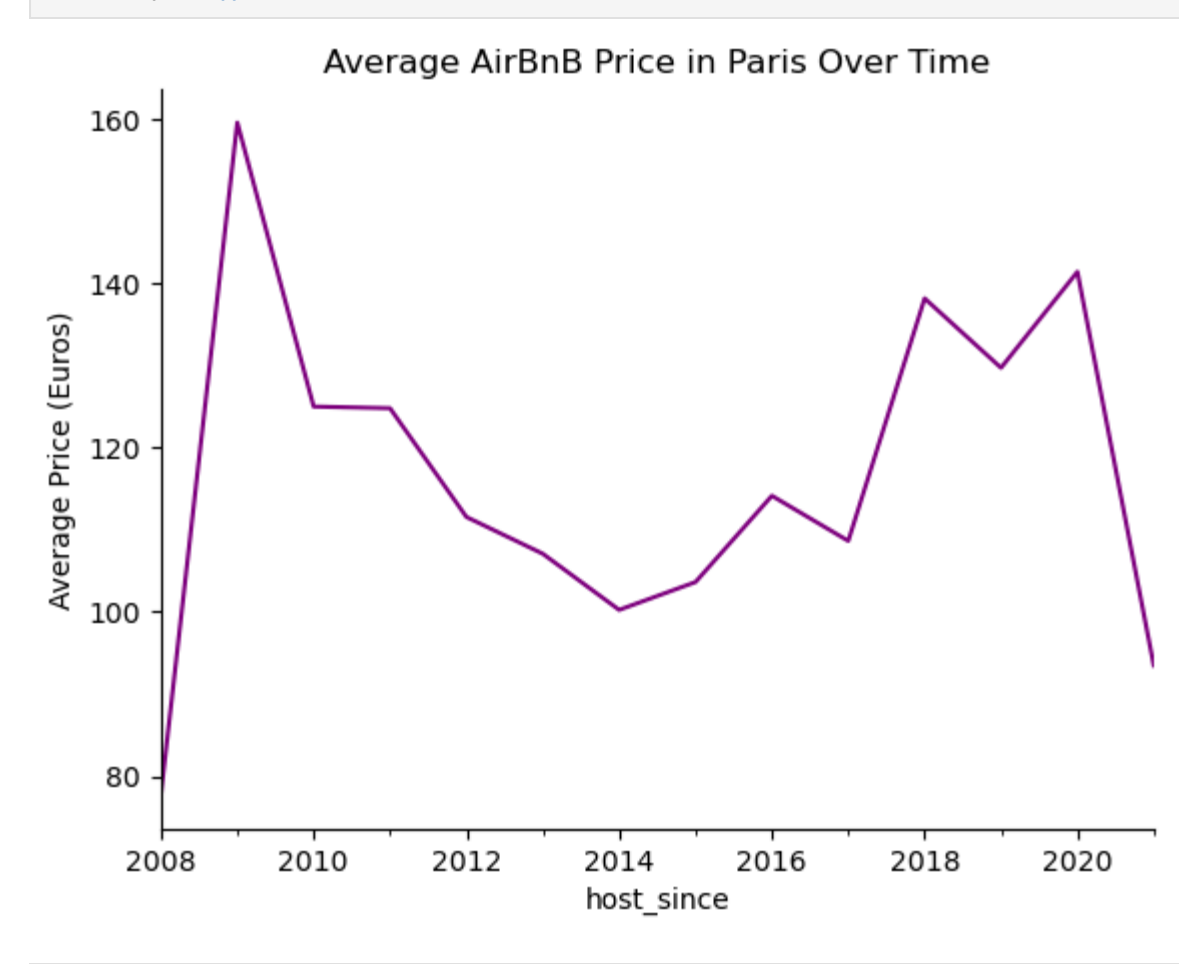


In [115…
```python
#Create a horizontal bar chart of the average price by ‘accommodates’ in Paris’ most expensive neighborhood,
#and make sure to add a title and change axis labels as needed



paris_df_accommodates.plot.barh(
    title = "Average Listing Price by Accommodation#",
    xlabel = "Price Per Night(Euros)",
    ylabel = "Accommodation Capacity",
    legend = None #the legend was "Price" and it is obvious that n X axis it is the Price.

)

sns.despine()
```

In [104…
```python
#Create two line charts: one showing the count of new hosts over time, and one showing average price.
#Set the y-axis limit to 0, add a title, and change axis labels as needed

#LINE CHART ONE: COUNT OF HOSTS

paris_df_over_time["neighbourhood"].plot(
    ylabel = "New Hosts",
    title = "New AirBnB Hosts in Paris Over Time",
    c = "grey"

)

sns.despine()
```



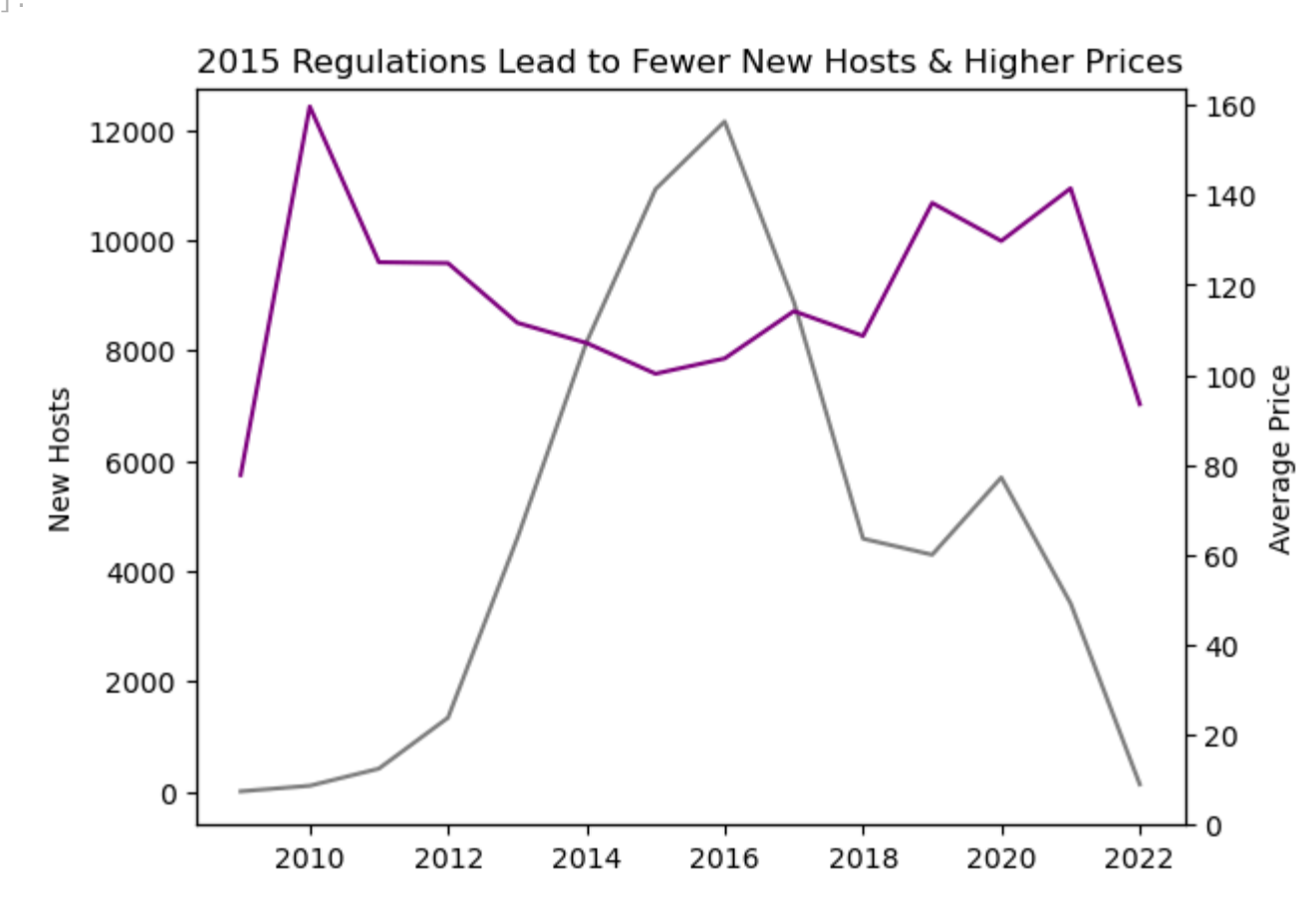
In [103…
```python
#LINE CHART TWO: AVG PRICE

paris_df_over_time["price"].plot(
    ylabel = "Average Price (Euros)",
    title = "Average AirBnB Price in Paris Over Time",
    c = "purple"

)

sns.despine()
```



In [89]:
```python
#Create a dual axis line chart to show the above both new hosts and average price over time in one chart.

import matplotlib.pyplot as plt
```

In [112…
```python
fig, ax = plt.subplots()

ax.plot(
    paris_df_over_time.index,
    paris_df_over_time["neighbourhood"],
    label = "New Host",
    c="grey"
)

ax.set_ylabel("New Hosts") #till here we have ploted the first chart.

#Below we will continue and plot the secont chart

ax2 = ax.twinx() #The twinx() in Python's Matplotlib library is used to create a second axes that shares the same x-axis as an existing axes object.

ax2.plot(
    paris_df_over_time.index,
    paris_df_over_time["price"],
    label = "Average Price",
    c="purple"
)

ax2.set_ylim(0)

ax2.set_ylabel("Average Price")

ax.set_title("2015 Regulations Lead to Fewer New Hosts & Higher Prices")
```

Out[112]: `Text(0.5, 1.0, '2015 Regulations Lead to Fewer New Hosts & Higher Prices')`



In [ ]:

In [ ]: