# Data Analysis and Regression Model - An Insight into Airbnb Listing Data

# Contents

# Introduction

This presentation provides an in-depth look at a comprehensive data analysis and regression modeling process applied to Airbnb listing data. Through the utilization of Python's powerful libraries such as Pandas and Scikit-learn, we'll explore data preprocessing, exploratory data analysis, feature engineering, and the development and evaluation of several regression models to predict listing prices. Our objective is to demonstrate the application of data science techniques in real-world scenarios, specifically in optimizing Airbnb listing prices for better market performance.

# Data Overview

## Data Sources

The analysis utilizes four key datasets: 'calendar.csv', 'reviews.csv', 'listings.csv', and 'hosts.csv'. These datasets encompass details on calendar listings, customer reviews, property listings, and host information respectively.

## Initial Data Processing

Data pre-processing steps include merging different datasets based on 'listing_id' and 'host_id', generating aggregated views, and handling missing values and data types.

# Exploratory Data Analysis (EDA)

## Key Insights

EDA reveals insights on price distribution, the relationship between 'available' and 'price', and correlations among variables.

## Data Quality Reporting

Continuous and categorical variables were analyzed to identify missing values, unique counts, and data anomalies. Anomalies were addressed to ensure data quality.

# Feature Engineering

## One-hot Encoding

Categorical variables like 'property_type', 'room_type', and 'bathrooms_text' were one-hot encoded to transform them into a format suitable for modeling.

## Log Transformation

A log transformation was applied to the 'price' variable to normalize its distribution, improving model's performance.

# Regression Model Development

## Model Selection

Various regression models including Linear Regression, Decision Tree, Random Forest, and Gradient Boosting were developed and evaluated.

## Model Evaluation

R-squared scores for each model were calculated. The data indicated potential overfitting in complex models, leading to the selection of Linear Regression as the preferred model.

# Conclusion

The study demonstrates the power of data analysis and machine learning in understanding and optimizing Airbnb listing prices. Through careful data preprocessing, exploratory data analysis, and methodical model evaluation, we can achieve significant insights into price optimization strategies. This process highlights the importance of data-driven decision-making in the competitive Airbnb market.

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
df1 = pd.read_csv('calendar.csv')
df2 = pd.read_csv('reviews.csv')
df3 = pd.read_csv('listings.csv')
df4 = pd.read_csv('hosts.csv')
```

In [4]:
```
df1.head()
```

Out[4]:

| | calender_id | listing_id | date | available | price | adjusted_price | minimum_nights | maximum_nights |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 40334325 | 2022-08-03 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 |
| **1** | 2 | 22742449 | 2022-11-13 00:00:00.000000 | 1 | 95.0 | 95.0 | 2 | 99 |
| **2** | 3 | 34621717 | 2022-04-17 00:00:00.000000 | 0 | 75.0 | 75.0 | 2 | 1125 |
| **3** | 4 | 38281744 | 2022-01-31 00:00:00.000000 | 1 | 150.0 | 150.0 | 1 | 1000 |
| **4** | 5 | 18835003 | 2022-05-21 00:00:00.000000 | 0 | 100.0 | 100.0 | 2 | 1125 |

In [5]:
```
df2.head()
```

Out[5]:

| | review_id | listing_id | date | reviewer_id | reviewer_name | comments |
|---|---|---|---|---|---|---|
| **0** | 1 | 50904 | 2015-05-06 00:00:00.000000 | 19482395 | Jihae | Karin's "Aplace" is absolutely beautiful and c... |
| **1** | 2 | 50904 | 2021-10-10 00:00:00.000000 | 333559 | Emilie | Karin is a wonderful host, she was really help... |
| **2** | 3 | 116134 | 2012-03-05 00:00:00.000000 | 928644 | Aurélien | Amazing flat, really close from the MAS Musem,... |
| **3** | 4 | 116134 | 2012-05-25 00:00:00.000000 | 231288 | Gail | This is a well equipped, very comfortable apar... |
| **4** | 5 | 116134 | 2013-09-03 00:00:00.000000 | 7984251 | Marcel | This is a very nice appartement. We really lik... |

In [6]:
```
df3.head()
```

Out[6]:

| | listing_id | listing_url | name | description | latitude | longitude | property_type | room_type | accomodates | bathrooms_text | bedrooms | beds | an |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 50904 | https://www.airbnb.com/rooms/50904 | aplace/ antwerp: cosy suite - fashion district | Decorated in a vintage style combined with a f... | 51.218575 | 4.398631 | Room in boutique hotel | Hotel room | 2 | 1 private bath | 1.0 | 1.0 | ["K alarr dryer", |
| 1 | 116134 | https://www.airbnb.com/rooms/116134 | Spacious apartment nearby Mas | Enjoy your stay at our 4 person apartment in t... | 51.230510 | 4.405930 | Entire rental unit | Entire home/apt | 4 | 2.5 baths | 2.0 | 2.0 | ["Refrig "E "Pa |
| 2 | 218916 | https://www.airbnb.com/rooms/218916 | Apartment with terrace in trendy Zurenborg | Do you enjoy authentic places with a lot of ch... | 51.206330 | 4.429420 | Entire condominium (condo) | Entire home/apt | 5 | 1 bath | 1.0 | 3.0 | ["Pa pa pre "Ki |
| 3 | 224333 | https://www.airbnb.com/rooms/224333 | Large stylish room in 1930s house + garden | Large bedroom in classic 1930s house. Kitchen,... | 51.197720 | 4.458530 | Private room in residential home | Private room | 2 | 2 shared baths | 1.0 | 1.0 | exting "Ba "Long |
| 4 | 224682 | https://www.airbnb.com/rooms/224682 | APARTMENT ROSCAM - OLD CENTRE ANTWERP | The space<br />Apartment "Roscam" is a ... | 51.217220 | 4.397900 | Entire rental unit | Entire home/apt | 3 | 1 bath | 1.0 | 2.0 | ["Refrig "K alarn |

In [7]:
```
df4.head()
```

Out[7]:

| | host_id | host_name | host_since | host_location | host_about |
|---|---|---|---|---|---|
| 0 | 234077 | Karin | 2010-09-14 00:00:00.000000 | Antwerp, Flanders, Belgium | Ever since my childhood I dreamt of having my ... |
| 1 | 334804 | Ann | 2011-01-04 00:00:00.000000 | Antwerp, Flemish Region, Belgium | Ciao, \r\nlooking forward to meet you!\r\nI lo... |
| 2 | 413052 | Valérie | 2011-02-27 00:00:00.000000 | Antwerp, Flanders, Belgium | NaN |
| 3 | 452791 | Tatiana | 2011-03-20 00:00:00.000000 | Antwerp, Flanders, Belgium | World traveler with a penchant for adrenaline ... |
| 4 | 462975 | Els | 2011-03-25 00:00:00.000000 | Edegem, Flanders, Belgium | I studied languages and cultural anthropology ... |

In [8]:
```
merge1 = pd.merge(df1,df3,how = 'inner', on = 'listing_id')
```

In [9]:
```
merge2 = pd.merge(merge1,df4,how = 'inner', on = 'host_id')
```

In [10]:
```
merge2.head()
```

Out[10]:

| | calender_id | listing_id | date | available | price | adjusted_price | minimum_nights | maximum_nights | listing_url | name | ... | accomodates | ba |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 40334325 | 2022-08-03 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 2 | |
| 1 | 2109 | 40334325 | 2022-02-14 00:00:00.000000 | 1 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 2 | |

In [11]:
```
merge1.shape
```

Out[11]:
```
(319192, 21)
```

In [12]:
```
df4.shape
```

Out[12]:
```
(1111, 5)
```

In [13]:
```
merge2.shape
```

Out[13]:
```
(319192, 25)
```

In [14]:
```
merge2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319192 entries, 0 to 319191
Data columns (total 25 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   calender_id     319192 non-null  int64
 1   listing_id      319192 non-null  int64
 2   date            319192 non-null  object
 3   available       319192 non-null  int64
 4   price           319117 non-null  float64
 5   adjusted_price  319117 non-null  float64
 6   minimum_nights  319192 non-null  int64
 7   maximum_nights  319192 non-null  int64
 8   listing_url     319192 non-null  object
 9   name            319192 non-null  object
 10  description     306489 non-null  object
```

```
 1   listing_id       319192 non-null  int64
 2   date             319192 non-null  object
 3   available        319192 non-null  int64
 4   price            319117 non-null  float64
 5   adjusted_price   319117 non-null  float64
 6   minimum_nights   319192 non-null  int64
 7   maximum_nights   319192 non-null  int64
 8   listing_url      319192 non-null  object
 9   name             319192 non-null  object
 10  description      306489 non-null  object
 11  latitude         319192 non-null  float64
 12  longitude        319192 non-null  float64
 13  property_type    319192 non-null  object
 14  room_type        319192 non-null  object
 15  accomodates      319192 non-null  int64
 16  bathrooms_text   319192 non-null  object
 17  bedrooms         295816 non-null  float64
 18  beds             311764 non-null  float64
 19  amenities        319192 non-null  object
 20  host_id          319192 non-null  int64
 21  host_name        319192 non-null  object
 22  host_since       319192 non-null  object
 23  host_location    318631 non-null  object
 24  host_about       157616 non-null  object
dtypes: float64(6), int64(7), object(12)
memory usage: 60.9+ MB
```

In [15]:
```
final_df = pd.merge(merge2,df3,how = 'inner', on = 'listing_id')
```

In [16]:
```
final_df.columns
```

Out[16]:
```
Index(['calender_id', 'listing_id', 'date', 'available', 'price',
       'adjusted_price', 'minimum_nights', 'maximum_nights', 'listing_url_x',
       'name_x', 'description_x', 'latitude_x', 'longitude_x',
       'property_type_x', 'room_type_x', 'accomodates_x', 'bathrooms_text_x',
       'bedrooms_x', 'beds_x', 'amenities_x', 'host_id_x', 'host_name',
       'host_since', 'host_location', 'host_about', 'listing_url_y', 'name_y',
       'description_y', 'latitude_y', 'longitude_y', 'property_type_y',
       'room_type_y', 'accomodates_y', 'bathrooms_text_y', 'bedrooms_y',
       'beds_y', 'amenities_y', 'host_id_y'],
      dtype='object')
```

## Create an aggregated view of data spread across different tables, containing the target as well as predictor variables.

In [17]:
```
final_df.head()
```

Out[17]:

| | calender_id | listing_id | date | available | price | adjusted_price | minimum_nights | maximum_nights | listing_url_x | name_x | ... | latitude_y | longitu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 40334325 | 2022-08-03 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| **1** | 2109 | 40334325 | 2022-02-14 00:00:00.000000 | 1 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| **2** | 3617 | 40334325 | 2022-04-26 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| **3** | 5560 | 40334325 | 2022-04-08 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| **4** | 8188 | 40334325 | 2022-04-11 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |

## Look at the table Calendar how many rows and unique listing ids are present? Are there any implications when it comes to having more rows and less unique listing ids?

In [18]:
```
unique_listing_ids = df1['listing_id'].nunique()
```

In [19]:
```
print(unique_listing_ids)
```

```
1749
```

We can consider listing_id as a primary key

## Look at the price column in Calendar table. What transformations you will need to perform so that you can create a column that can be used as a target/ response variable?

In [20]:
```
df1.isna().sum()
```

Out[20]:
```
calender_id        0
listing_id         0
date               0
available          0
price             75
adjusted_price    75
minimum_nights     0
maximum_nights     0
dtype: int64
```

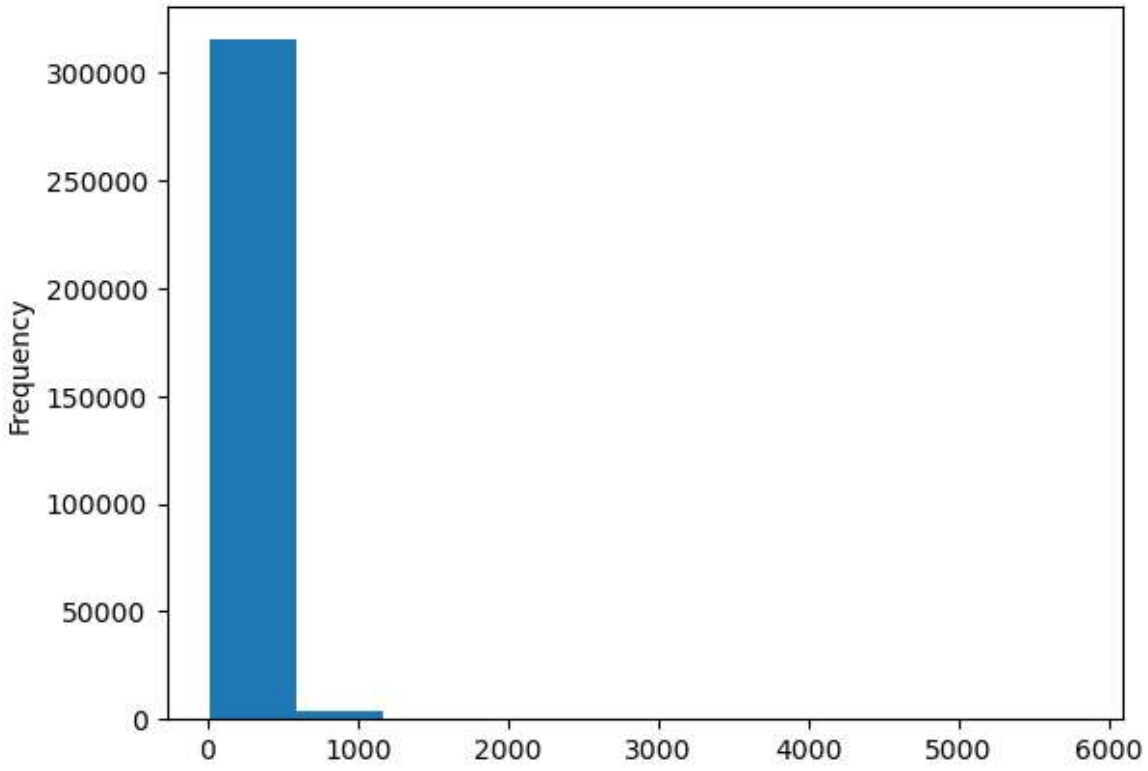
In [21]:
```
df1.describe()
```

Out[21]:

| | calender_id | listing_id | available | price | adjusted_price | minimum_nights | maximum_nights |
|---|---|---|---|---|---|---|---|
| **count** | 319192.000000 | 3.191920e+05 | 319192.000000 | 319117.000000 | 319117.000000 | 319192.000000 | 319192.000000 |
| **mean** | 159596.500000 | 3.488528e+07 | 0.535192 | 109.917779 | 109.712131 | 5.379395 | 812.206102 |
| **std** | 92142.937899 | 1.523257e+07 | 0.498761 | 185.791168 | 185.551851 | 21.456127 | 511.622075 |
| **min** | 1.000000 | 5.090400e+04 | 0.000000 | 13.000000 | 13.000000 | 1.000000 | 1.000000 |
| **25%** | 79798.750000 | 2.338661e+07 | 0.000000 | 59.000000 | 58.000000 | 1.000000 | 365.000000 |
| **50%** | 159596.500000 | 3.891969e+07 | 1.000000 | 79.000000 | 79.000000 | 2.000000 | 1125.000000 |
| **75%** | 239394.250000 | 4.839174e+07 | 1.000000 | 115.000000 | 115.000000 | 3.000000 | 1125.000000 |
| **max** | 319192.000000 | 5.398332e+07 | 1.000000 | 5800.000000 | 5800.000000 | 500.000000 | 9999.000000 |

In [22]:
```
df1['price'].plot(kind = 'hist')
```

Out[22]: <Axes: ylabel='Frequency'>



In [23]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319192 entries, 0 to 319191
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   calender_id     319192 non-null  int64
 1   listing_id      319192 non-null  int64
 2   date            319192 non-null  object
 3   available       319192 non-null  int64
 4   price           319117 non-null  float64
 5   adjusted_price  319117 non-null  float64
 6   minimum_nights  319192 non-null  int64
 7   maximum_nights  319192 non-null  int64
dtypes: float64(2), int64(5), object(1)
memory usage: 19.5+ MB
```

In [24]:
```
final_df['price'].fillna(df1['price'].mean(), inplace=True)
```

we have imputed mean for the missing values in price coloumn

## Look at the tables Listings, Hosts and Reviews to come up with a list of potential transformations needed in order to have predictors that can be used to predict the listing price.

In [25]:
```
final_df.head()
```

Out[25]:

| | calender_id | listing_id | date | available | price | adjusted_price | minimum_nights | maximum_nights | listing_url_x | name_x | ... | latitude_y | longitu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 40334325 | 2022-08-03 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| 1 | 2109 | 40334325 | 2022-02-14 00:00:00.000000 | 1 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| 2 | 3617 | 40334325 | 2022-04-26 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| 3 | 5560 | 40334325 | 2022-04-08 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/40334325 | Luxurious flat in central location | ... | 51.20989 | 4.4 |
| 4 | 8188 | 40334325 | 2022-04-11 00:00:00.000000 | 0 | 56.0 | 56.0 | 3 | 5 | https://www.airbnb.com/rooms/ | Luxurious flat in central | ... | 51.20989 | 4.4 |

In [26]:
```
final_df.isna().sum()
```

Out[26]:

| | |
|---|---|
| calender_id | 0 |
| listing_id | 0 |
| date | 0 |
| available | 0 |
| price | 0 |
| adjusted_price | 75 |
| minimum_nights | 0 |
| maximum_nights | 0 |
| listing_url_x | 0 |
| name_x | 0 |
| description_x | 12703 |
| latitude_x | 0 |
| longitude_x | 0 |
| property_type_x | 0 |
| room_type_x | 0 |
| accomodates_x | 0 |
| bathrooms_text_x | 0 |
| bedrooms_x | 23376 |
| beds_x | 7428 |
| amenities_x | 0 |
| host_id_x | 0 |
| host_name | 0 |
| host_since | 0 |
| host_location | 561 |
| host_about | 161576 |
| listing_url_y | 0 |
| name_y | 0 |

In [27]:
```
columns_to_remove = ['adjusted_price', 'listing_url_x', 'name_x', 'description_x', 'latitude_x', 'longitude_x',
                     'bedrooms_x', 'beds_x', 'host_name', 'host_location', 'host_about', 'listing_url_y',
                     'name_y', 'description_y', 'latitude_y', 'longitude_y', 'property_type_y', 'room_type_y',
                     'accomodates_y', 'bathrooms_text_y', 'bedrooms_y', 'beds_y', 'amenities_y', 'host_id_y','amenities_x']
final_df.drop(columns=columns_to_remove, inplace=True)
```

In [28]:
```
final_df.describe()
```

Out[28]:

| | calender_id | listing_id | available | price | minimum_nights | maximum_nights | accomodates_x | host_id_x |
|---|---|---|---|---|---|---|---|---|
| **count** | 319192.000000 | 3.191920e+05 | 319192.000000 | 319192.000000 | 319192.000000 | 319192.000000 | 319192.000000 | 3.191920e+05 |
| **mean** | 159596.500000 | 3.488528e+07 | 0.535192 | 109.917779 | 5.379395 | 812.206102 | 3.762619 | 1.418691e+08 |
| **std** | 92142.937899 | 1.523257e+07 | 0.498761 | 185.769339 | 21.456127 | 511.622075 | 2.771459 | 1.287545e+08 |
| **min** | 1.000000 | 5.090400e+04 | 0.000000 | 13.000000 | 1.000000 | 1.000000 | 1.000000 | 2.340770e+05 |
| **25%** | 79798.750000 | 2.338661e+07 | 0.000000 | 59.000000 | 1.000000 | 365.000000 | 2.000000 | 2.875771e+07 |
| **50%** | 159596.500000 | 3.891969e+07 | 1.000000 | 79.000000 | 2.000000 | 1125.000000 | 3.000000 | 1.033579e+08 |
| **75%** | 239394.250000 | 4.839174e+07 | 1.000000 | 115.000000 | 3.000000 | 1125.000000 | 4.000000 | 2.354916e+08 |
| **max** | 319192.000000 | 5.398332e+07 | 1.000000 | 5800.000000 | 500.000000 | 9999.000000 | 16.000000 | 4.373093e+08 |

In [29]:
```
final_df.shape
```

Out[29]:
```
(319192, 13)
```

In [30]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319192 entries, 0 to 319191
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   calender_id       319192 non-null  int64  
 1   listing_id        319192 non-null  int64  
 2   date              319192 non-null  object 
 3   available         319192 non-null  int64  
 4   price             319192 non-null  float64
 5   minimum_nights    319192 non-null  int64  
 6   maximum_nights    319192 non-null  int64  
 7   property_type_x   319192 non-null  object 
 8   room_type_x       319192 non-null  object 
 9   accomodates_x     319192 non-null  int64  
 10  bathrooms_text_x  319192 non-null  object 
 11  host_id_x         319192 non-null  int64  
 12  host_since        319192 non-null  object 
dtypes: float64(1), int64(7), object(5)
memory usage: 31.7+ MB
```

In [31]:
```
final_df.isna().sum()
```

Out[31]:
```
calender_id          0
listing_id           0
date                 0
available            0
price                0
minimum_nights       0
maximum_nights       0
property_type_x      0
room_type_x          0
accomodates_x        0
bathrooms_text_x     0
host_id_x            0
host_since           0
dtype: int64
```

In [32]:
```
final_df.head()
```

Out[32]:

| | calender_id | listing_id | date | available | price | minimum_nights | maximum_nights | property_type_x | room_type_x | accomodates_x | bathrooms_text_ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 40334325 | 2022-08-03 00:00:00.000000 | 0 | 56.0 | 3 | 5 | Entire rental unit | Entire home/ apt | 2 | 1 ba |
| **1** | 2109 | 40334325 | 2022-02-14 00:00:00.000000 | 1 | 56.0 | 3 | 5 | Entire rental unit | Entire home/ apt | 2 | 1 ba |
| **2** | 3617 | 40334325 | 2022-04-26 00:00:00.000000 | 0 | 56.0 | 3 | 5 | Entire rental unit | Entire home/ apt | 2 | 1 ba |
| **3** | 5560 | 40334325 | 2022-04-08 00:00:00.000000 | 0 | 56.0 | 3 | 5 | Entire rental unit | Entire home/ apt | 2 | 1 ba |
| **4** | 8188 | 40334325 | 2022-04-11 00:00:00.000000 | 0 | 56.0 | 3 | 5 | Entire rental unit | Entire home/ apt | 2 | 1 ba |

• Once the aggregated dataset has been created, do a data audit. Create a data quality report which has the following basic structure: • Continuous Variables: (#unique values, percentage_missing_values, min, max, average, 25th percentile, 75th percentile, 90th percentile, 95th percentile) • Categorical Variables: (#Unique values, percentage_missing_values) • Highlight any data anomaly that you find and fix it.

In [33]:
```python
def data_quality_report(df):
    report = {}
    continuous_vars = df.select_dtypes(include=['float64']).columns
    continuous_report = df[continuous_vars].describe(percentiles=[.25, .75, .90, .95])
    continuous_report['#unique_values'] = df[continuous_vars].nunique()
    continuous_report['percentage_missing_values'] = df[continuous_vars].isnull().mean() * 100

    report['Continuous Variables'] = continuous_report


    categorical_vars = df.select_dtypes(include=['object']).columns
    categorical_report = pd.DataFrame(index=categorical_vars, columns=['#unique_values', 'percentage_missing_values'])
    for col in categorical_vars:
        categorical_report.loc[col, '#unique_values'] = df[col].nunique()
        categorical_report.loc[col, 'percentage_missing_values'] = df[col].isnull().mean() * 100

    report['Categorical Variables'] = categorical_report

    return report


quality_report = data_quality_report(final_df)


for var_type, var_report in quality_report.items():
    print(var_type)
    print(var_report)
    print()
```

Continuous Variables

| | price | #unique_values | percentage_missing_values |
|---|---|---|---|
| count | 319192.000000 | NaN | NaN |
| mean | 109.917779 | NaN | NaN |
| std | 185.769339 | NaN | NaN |
| min | 13.000000 | NaN | NaN |
| 25% | 59.000000 | NaN | NaN |
| 50% | 79.000000 | NaN | NaN |
| 75% | 115.000000 | NaN | NaN |
| 90% | 180.000000 | NaN | NaN |
| 95% | 250.000000 | NaN | NaN |
| max | 5800.000000 | NaN | NaN |

Categorical Variables

| | #unique_values | percentage_missing_values |
|---|---|---|
| date | 365 | 0.0 |
| property_type_x | 39 | 0.0 |
| room_type_x | 4 | 0.0 |
| bathrooms_text_x | 26 | 0.0 |
| host_since | 927 | 0.0 |

In [34]:
```
final_df.dtypes
```

Out[34]:
```
calender_id           int64
listing_id            int64
date                 object
available             int64
price               float64
minimum_nights        int64
maximum_nights        int64
property_type_x      object
room_type_x          object
accomodates_x         int64
bathrooms_text_x     object
host_id_x             int64
host_since           object
dtype: object
```

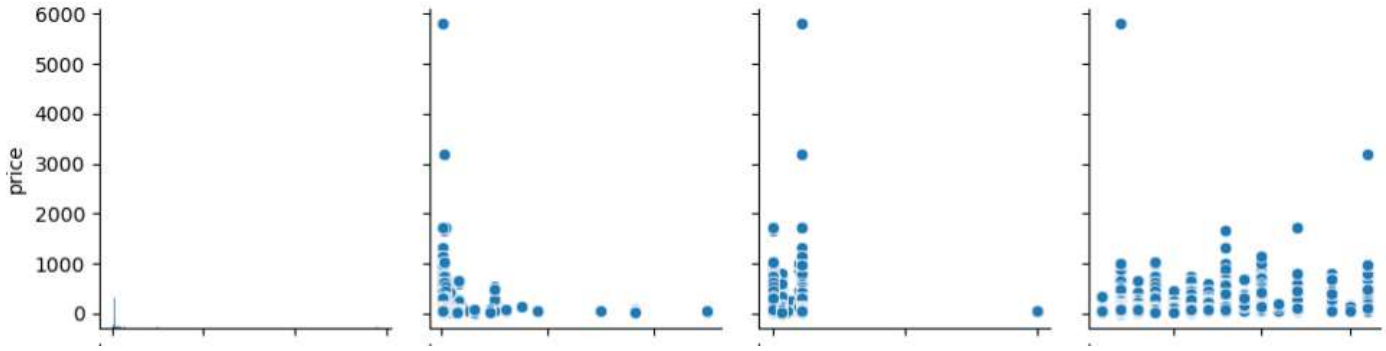
In [38]:
```
import seaborn as sns
import matplotlib.pyplot as plt

final_df['date'] = pd.to_datetime(final_df['date'])
final_df['host_since'] = pd.to_datetime(final_df['host_since'])
```

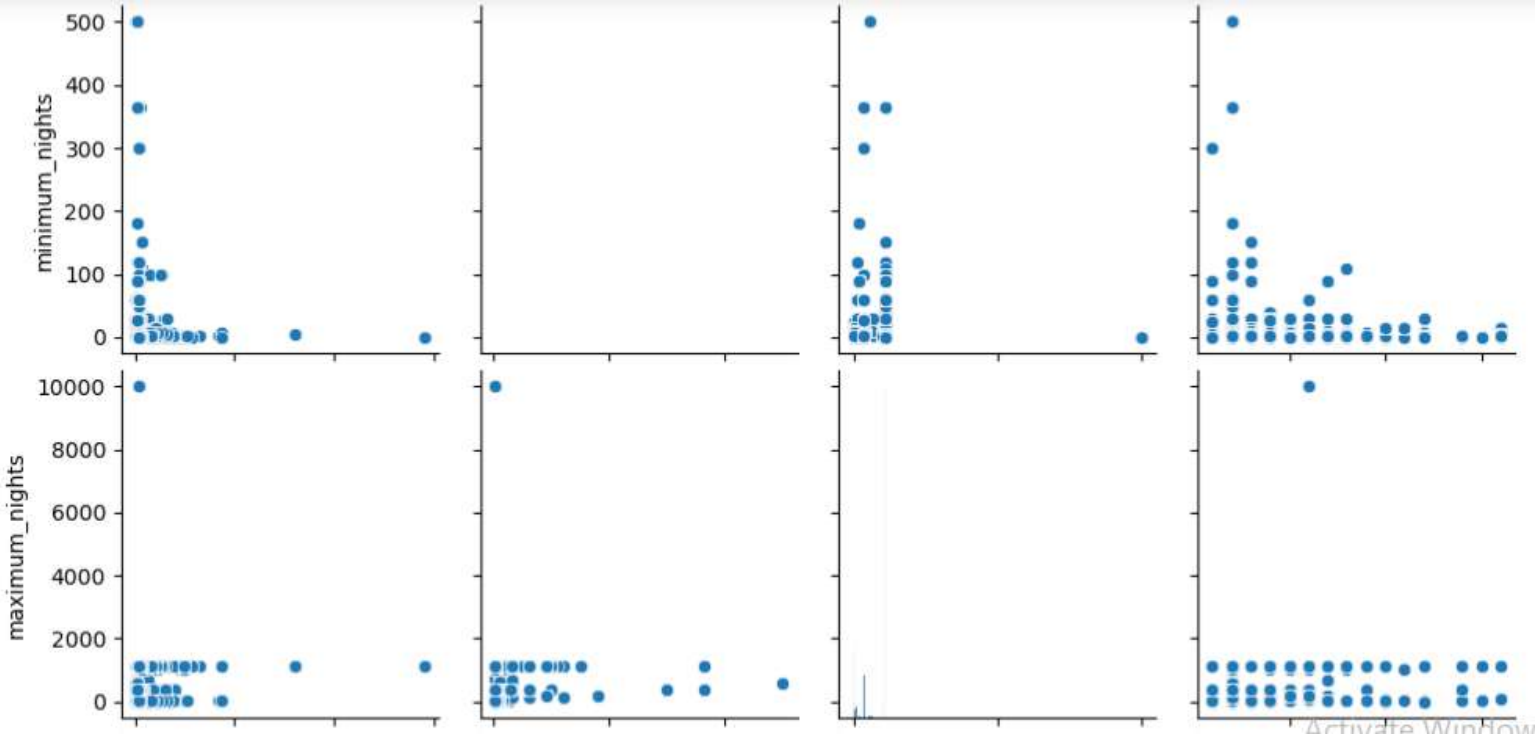
In [42]:
```
correlation_matrix = final_df.corr(numeric_only = True)
print(correlation_matrix['price'])
```

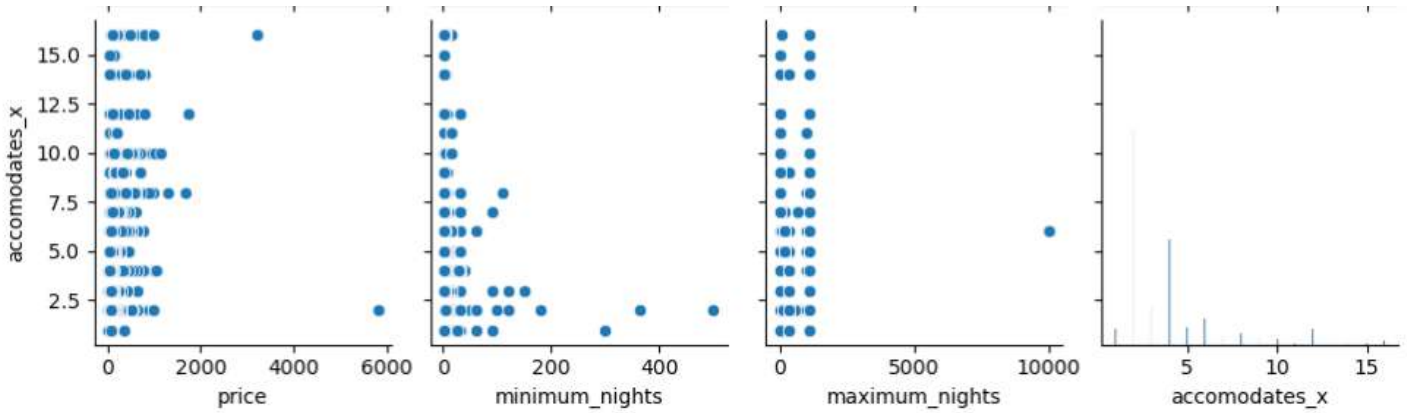
```
calender_id       0.000029
listing_id        0.025577
available         0.019126
price             1.000000
minimum_nights   -0.012969
maximum_nights    0.041521
accomodates_x     0.199957
host_id_x         0.064113
Name: price, dtype: float64
```

In [43]:
```
sns.pairplot(final_df, vars=['price', 'minimum_nights', 'maximum_nights', 'accomodates_x'])
plt.show()
```

```
C:\Users\Personal\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




minimum_nights
500
400
300
200
100
0
maximum_nights
10000
8000
6000
4000
2000
0


accomodates_x
15.0
12.5
10.0
7.5
5.0
2.5
0
2000
4000
6000
price
0
200
400
minimum_nights
0
5000
10000
maximum_nights
5
10
15
accomodates_x

In [44]:
```
point_biserial_corr = final_df['available'].corr(final_df['price'])
print("Point-biserial correlation between 'available' and 'price':", point_biserial_corr)
```

```
Point-biserial correlation between 'available' and 'price': 0.019125606778687354
```

In [46]:
```
property_dummies = pd.get_dummies(final_df['property_type_x'])
room_dummies = pd.get_dummies(final_df['room_type_x'])
bathrooms_dummies = pd.get_dummies(final_df['bathrooms_text_x'])
```

In [47]:
```
property_dummies
```

Out[47]:

| | Boat | Casa particular | Castle | Entire condominium (condo) | Entire cottage | Entire guest suite | Entire guesthouse | Entire loft | Entire rental unit | Entire residential home | ... | Room in aparthotel | Room in boutique hotel | Room in hotel | Shared room in bed and breakfast | Shared room in casa particular | Sl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | True | False | ... | False | False | False | False | False | |
| **1** | False | False | False | False | False | False | False | False | True | False | ... | False | False | False | False | False | |
| **2** | False | False | False | False | False | False | False | False | True | False | ... | False | False | False | False | False | |
| **3** | False | False | False | False | False | False | False | False | True | False | ... | False | False | False | False | False | |
| **4** | False | False | False | False | False | False | False | False | True | False | ... | False | False | False | False | False | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **319187** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | |
| **319188** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | |
| **319189** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | |
| **319190** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | |
| **319191** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | |

319192 rows × 39 columns

In [50]:
```
# Concatenate new one-hot encoded columns with the original DataFrame
df = pd.concat((final_df.drop(['property_type_x', 'room_type_x', 'bathrooms_text_x'], axis=1),
                property_dummies.astype(int),
                room_dummies.astype(int),
                bathrooms_dummies.astype(int)),
                axis=1)

print('Number of Columns:', len(df.columns))

# Move the target variable 'price' to the end of the DataFrame
cols = list(df.columns.values)
idx = cols.index('price')
rearrange_cols = cols[:idx] + cols[idx+1:] + [cols[idx]]
df = df[rearrange_cols]

# Check the updated DataFrame
df.head()
```

```
Number of Columns: 79
```

Out[50]:

| | calender_id | listing_id | date | available | minimum_nights | maximum_nights | accomodates_x | host_id_x | host_since | Boat | ... | 5 baths | 6 baths | 6.5 baths | 7 bat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 40334325 | 2022-08-03 | 0 | 3 | 5 | 2 | 311556587 | 2019-11-23 | 0 | ... | 0 | 0 | 0 | |
| **1** | 2109 | 40334325 | 2022-02-14 | 1 | 3 | 5 | 2 | 311556587 | 2019-11-23 | 0 | ... | 0 | 0 | 0 | |
| **2** | 3617 | 40334325 | 2022-04-26 | 0 | 3 | 5 | 2 | 311556587 | 2019-11-23 | 0 | ... | 0 | 0 | 0 | |
| **3** | 5560 | 40334325 | 2022-04-08 | 0 | 3 | 5 | 2 | 311556587 | 2019-11-23 | 0 | ... | 0 | 0 | 0 | |
| **4** | 8188 | 40334325 | 2022-04-11 | 0 | 3 | 5 | 2 | 311556587 | 2019-11-23 | 0 | ... | 0 | 0 | 0 | |

5 rows × 81 columns

In [57]:
```
df.columns
```

Out[57]:
```
Index(['host_id_x', 'calender_id', 'listing_id', 'date', 'available', 'price',
       'minimum_nights', 'maximum_nights', 'accomodates_x', 'host_since',
       'Boat', 'Casa particular', 'Castle', 'Entire condominium (condo)',
       'Entire cottage', 'Entire guest suite', 'Entire guesthouse',
       'Entire loft', 'Entire rental unit', 'Entire residential home',
       'Entire serviced apartment', 'Entire townhouse', 'Entire vacation home',
       'Entire villa', 'Houseboat', 'Private room', 'Private room',
       'Private room', 'Private room', 'Private room in bed and breakfast',
       'Private room in boat', 'Private room in casa particular',
       'Private room in condominium (condo)', 'Private room in guest suite',
       'Private room in guesthouse', 'Private room in loft',
       'Private room in religious building', 'Private room in rental unit',
       'Private room in residential home',
       'Private room in serviced apartment', 'Private room in townhouse',
       'Private room in villa', 'Room in aparthotel', 'Room in boutique hotel',
       'Room in hotel', 'Shared room in bed and breakfast',
       'Shared room in casa particular', 'Shared room in loft',
       'Shared room in residential home', 'Tent', 'Tiny house', 'Yurt',
       'Entire home/apt', 'Hotel room', 'Private room', 'Private room',
       'Private room', 'Private room', 'Shared room', '0 baths',
       '0 shared baths', '1 bath', '1 private bath', '1 shared bath',
       '1.5 baths', '1.5 shared baths', '15.5 baths', '2 baths',
       '2 shared baths', '2.5 baths', '2.5 shared baths', '3 baths',
       '3 shared baths', '3.5 baths', '4 baths', '4 shared baths', '5 baths',
       '6 baths', '6.5 baths', '7.5 baths', '8 baths', '8.5 baths',
       'Half-bath', 'Private half-bath', 'Shared half-bath', 'log_price'],
      dtype='object')
```

In [53]:
```
# Get the index of 'host_id_x'
idx_host_id = cols.index('host_id_x')

# Move 'host_id_x' to the beginning of the list
rearrange_cols = [cols[idx_host_id]] + cols[:idx_host_id] + cols[idx_host_id+1:]

# Rearrange the DataFrame columns
df = df[rearrange_cols]

# Print the updated DataFrame
df.head()
```

Out[53]:

| | host_id_x | calender_id | listing_id | date | available | price | minimum_nights | maximum_nights | accomodates_x | host_since | ... | 4 shared baths | 5 baths | 6 baths | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 311556587 | 1 | 40334325 | 2022-08-03 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| **1** | 311556587 | 2109 | 40334325 | 2022-02-14 | 1 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| **2** | 311556587 | 3617 | 40334325 | 2022-04-26 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| **3** | 311556587 | 5560 | 40334325 | 2022-04-08 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| **4** | 311556587 | 8188 | 40334325 | 2022-04-11 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |

5 rows × 85 columns

In [54]:
```
df['price'] = df['price'].replace('[\$,)]','', regex=True).replace('[(]','-', regex=True).astype(float)
df['log_price'] = np.log(df['price'].values)
display(df.head())
df.columns
```

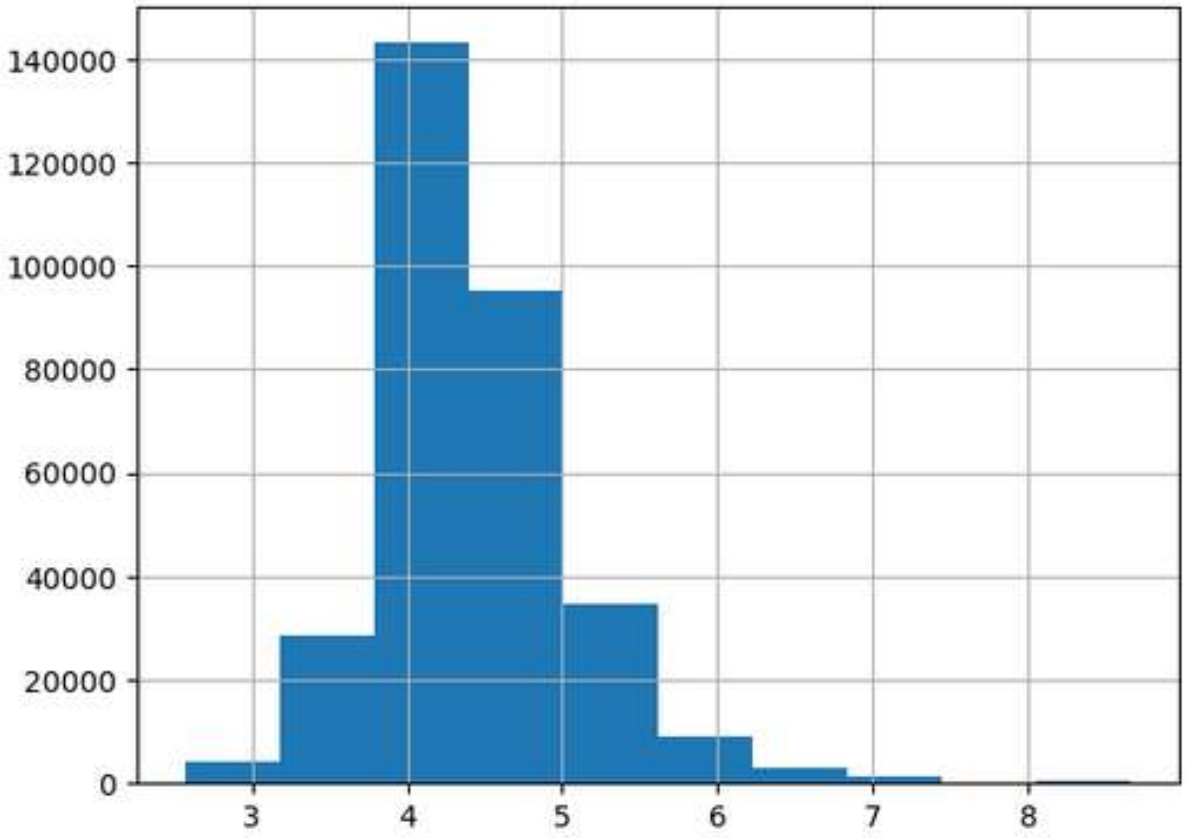
| | host_id_x | calender_id | listing_id | date | available | price | minimum_nights | maximum_nights | accomodates_x | host_since | ... | 5 baths | 6 baths | 6.5 baths | ba |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 311556587 | 1 | 40334325 | 2022-08-03 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| 1 | 311556587 | 2109 | 40334325 | 2022-02-14 | 1 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| 2 | 311556587 | 3617 | 40334325 | 2022-04-26 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| 3 | 311556587 | 5560 | 40334325 | 2022-04-08 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |
| 4 | 311556587 | 8188 | 40334325 | 2022-04-11 | 0 | 56.0 | 3 | 5 | 2 | 2019-11-23 | ... | 0 | 0 | 0 | |

5 rows × 86 columns

Out[54]:
```
Index(['host_id_x', 'calender_id', 'listing_id', 'date', 'available', 'price',
       'minimum_nights', 'maximum_nights', 'accomodates_x', 'host_since',
       'Boat', 'Casa particular', 'Castle', 'Entire condominium (condo)',
       'Entire cottage', 'Entire guest suite', 'Entire guesthouse',
       'Entire loft', 'Entire rental unit', 'Entire residential home',
       'Entire serviced apartment', 'Entire townhouse', 'Entire vacation home',
       'Entire villa', 'Houseboat', 'Private room', 'Private room',
       'Private room', 'Private room', 'Private room in bed and breakfast',
       'Private room in boat', 'Private room in casa particular',
       'Private room in condominium (condo)', 'Private room in guest suite',
       'Private room in guesthouse', 'Private room in loft',
       'Private room in religious building', 'Private room in rental unit',
       'Private room in residential home',
       'Private room in serviced apartment', 'Private room in townhouse',
       'Private room in villa', 'Room in aparthotel', 'Room in boutique hotel',
       'Room in hotel', 'Shared room in bed and breakfast',
       'Shared room in casa particular', 'Shared room in loft',
       'Shared room in residential home', 'Tent', 'Tiny house', 'Yurt',
       'Entire home/apt', 'Hotel room', 'Private room', 'Private room',
       'Private room', 'Private room', 'Shared room', '0 baths',
       '0 shared baths', '1 bath', '1 private bath', '1 shared bath',
       '1.5 baths', '1.5 shared baths', '15.5 baths', '2 baths',
       '2 shared baths', '2.5 baths', '2.5 shared baths', '3 baths',
       '3 shared baths', '3.5 baths', '4 baths', '4 shared baths', '5 baths',
       '6 baths', '6.5 baths', '7.5 baths', '8 baths', '8.5 baths',
       'Half-bath', 'Private half-bath', 'Shared half-bath', 'log_price'],
      dtype='object')
```

In [56]:
```
df.log_price.hist()
```

Out[56]: <Axes: >




Looks like a normal distribution

In [64]:
```
X = df.drop(['host_id_x', 'calender_id', 'listing_id', 'log_price','date','host_since'], axis=1).values
y = df['log_price'].values

# Verify the shape of X and y
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (319192, 80)
Shape of y: (319192,)
```

In [65]:
```
from sklearn.model_selection import train_test_split
import sklearn.metrics as sk

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# OLS regression
clf = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1)
clf.fit(x_train, y_train)
predicted = clf.predict(x_test)

# Calculate R2 score
score = sk.r2_score(y_test, predicted)
print('sklearn: R2 score for Linear Regression is: {}'.format(score))
```

```
sklearn: R2 score for Linear Regression is: 0.6173488357799001
```

In [66]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

# Decision Tree Regression
dt_regressor = DecisionTreeRegressor(random_state=42)
dt_regressor.fit(x_train, y_train)
dt_predicted = dt_regressor.predict(x_test)
dt_score = sk.r2_score(y_test, dt_predicted)
print('R2 score for Decision Tree Regression is: {}'.format(dt_score))

# Random Forest Regression
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(x_train, y_train)
rf_predicted = rf_regressor.predict(x_test)
rf_score = sk.r2_score(y_test, rf_predicted)
print('R2 score for Random Forest Regression is: {}'.format(rf_score))

# Gradient Boosting Regression
gbm_regressor = GradientBoostingRegressor(n_estimators=100, random_state=42)
gbm_regressor.fit(x_train, y_train)
gbm_predicted = gbm_regressor.predict(x_test)
gbm_score = sk.r2_score(y_test, gbm_predicted)
print('R2 score for Gradient Boosting Regression is: {}'.format(gbm_score))
```

```
R2 score for Decision Tree Regression is: 0.9999995897773652
R2 score for Random Forest Regression is: 0.9999995821163237
R2 score for Gradient Boosting Regression is: 0.9999638494939526
```

In [67]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

# Decision Tree Regression with regularization
dt_regressor = DecisionTreeRegressor(max_depth=5, random_state=42)  # Example regularization parameter: max_depth
dt_regressor.fit(x_train, y_train)
dt_predicted = dt_regressor.predict(x_test)
dt_score = sk.r2_score(y_test, dt_predicted)
print('R2 score for Decision Tree Regression with regularization is: {}'.format(dt_score))

# Random Forest Regression with regularization
rf_regressor = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)  # Example regularization parameter: max_
rf_regressor.fit(x_train, y_train)
rf_predicted = rf_regressor.predict(x_test)
rf_score = sk.r2_score(y_test, rf_predicted)
print('R2 score for Random Forest Regression with regularization is: {}'.format(rf_score))

# Gradient Boosting Regression with regularization
gbm_regressor = GradientBoostingRegressor(n_estimators=100, max_depth=5, random_state=42)  # Example regularization parameter:
gbm_regressor.fit(x_train, y_train)
gbm_predicted = gbm_regressor.predict(x_test)
gbm_score = sk.r2_score(y_test, gbm_predicted)
print('R2 score for Gradient Boosting Regression with regularization is: {}'.format(gbm_score))
```

```
R2 score for Decision Tree Regression with regularization is: 0.9977042886984263
R2 score for Random Forest Regression with regularization is: 0.9981654719808398
R2 score for Gradient Boosting Regression with regularization is: 0.9999979266002506
```



**Our Decision Tree Regression, Random Forest Regression & Gradient Boosting Regression are getting overfitted. So, we have choosen Linear Regression**