```
# Do NOT modify this code cell

# importing useful libraries
import re

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# Do NOT modify this code cell

# laoding in the data
df = pd.read_csv("https://raw.githubusercontent.com/MIE223-2024/course-datasets/main/listings.cs
print(df.shape)
df.head()
```

```
(20386, 18)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	1419	Home in Toronto · ★5.0 · 5 bedrooms · 7 beds ·...	1565	Alexandra	NaN	Little Portugal	43.64590	-79.42423	Ent home/a
1	8077	Rental unit in Toronto · ★4.84 · 1 bedroom · 1...	22795	Kathie & Larry	NaN	Waterfront Communities-The Island	43.64080	-79.37673	Priva roo
2	26654	Condo in Toronto · ★4.79 · 1 bedroom · 2 beds ...	113345	Adela	NaN	Waterfront Communities-The Island	43.64608	-79.39032	Ent home/a
3	27423	Rental unit in Toronto · ★4.93 · Studio · 1 be...	118124	Brent	NaN	South Riverdale	43.66884	-79.32725	Ent home/a
4	30931	Rental unit in Toronto · 1 bedroom · 2 beds · ...	22795	Kathie & Larry	NaN	Waterfront Communities-The Island	43.64015	-79.37625	Ent home/a

Creating `rating` column from `name` column by pulling all numbers after the ★ character.

```
# Do NOT modify this code cell

# creating 'rating' column
star_char = "★"

df['rating'] = df['name'].apply(lambda x: re.findall(star_char + "(\d+.?\d*)", x)[0] if len(re.findall(star_
df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	1419	Home in Toronto · ★5.0 · 5 bedrooms · 7 beds ·...	1565	Alexandra	NaN	Little Portugal	43.64590	-79.42423	Ent home/a
1	8077	Rental unit in Toronto · ★4.84 · 1 bedroom · 1...	22795	Kathie & Larry	NaN	Waterfront Communities-The Island	43.64080	-79.37673	Priva roo
2	26654	Condo in Toronto · ★4.79 · 1 bedroom · 2 beds ...	113345	Adela	NaN	Waterfront Communities-The Island	43.64608	-79.39032	Ent home/a
3	27423	Rental unit in Toronto · ★4.93 · Studio · 1 be...	118124	Brent	NaN	South Riverdale	43.66884	-79.32725	Ent home/a
4	30931	Rental unit in Toronto · 1 bedroom · 2 beds · ...	22795	Kathie & Larry	NaN	Waterfront Communities-The Island	43.64015	-79.37625	Ent home/a

Q1(a)

```
# your code starts here #

#df.types to get the data types of the columns
col_dt = df.dtypes

#to display the data types
col_dt

# end #
```

```
id                                 int64
name                              object
host_id                            int64
host_name                         object
neighbourhood_group              float64
neighbourhood                     object
latitude                         float64
longitude                        float64
room_type                         object
price                            float64
minimum_nights                     int64
number_of_reviews                  int64
last_review                       object
reviews_per_month                float64
calculated_host_listings_count     int64
availability_365                   int64
number_of_reviews_ltm              int64
license                           object
rating                            object
dtype: object
```

Q1(b)

```
# your code starts here #

#converting the column to float
df['rating'] = df['rating'].astype(np.float64)

#assigning the column to a variable for display
rating_dt = df['rating'].dtypes

#displaying the updated data type
rating_dt

# end #
```

```
dtype('float64')
```

+ Code + Text

Q2

```
# your code starts here #

#casting the data to float64 using the astype method
#describe method provides a statistical summary of the data
#the percentiles parameter specifies the required percentiles
statistics = df['rating'].astype(float).describe(percentiles=[.25, .5, .75])

#display the statistics
statistics

# end #
```

```
count    12137.000000
mean         4.789276
std          0.263310
min          1.670000
```

```
25%          4.710000
50%          4.860000
75%          4.970000
max          5.000000
Name: rating, dtype: float64
```

Your explanation goes here: Casting the data to float64 using the astype method. Describe method provides a statistical summary of the data.The percentiles parameter specifies the required percentiles

Q3(a)

```
# your code starts here

#isnull identifies the missing values in the df
#mean to claculate the mean of each col; mean is used as the cols are of boolean vals due to isnull
#sort_vlaues sorts the values in descending order
missing_percentages = df.isnull().mean().sort_values(ascending=False)

#display the output
missing_percentages

# end
```

```
neighbourhood_group               1.000000
license                           0.536152
rating                            0.404640
last_review                       0.257922
reviews_per_month                 0.257922
price                             0.185961
host_name                         0.000098
latitude                          0.000000
longitude                         0.000000
room_type                         0.000000
name                              0.000000
minimum_nights                    0.000000
number_of_reviews                 0.000000
neighbourhood                     0.000000
calculated_host_listings_count    0.000000
availability_365                  0.000000
number_of_reviews_ltm             0.000000
host_id                           0.000000
id                                0.000000
dtype: float64
```

Q3(b)

```
# your code starts here #

#the outer brackets is used for indexing
#the inner bracket is for the boolean series created by isnull
missing_host_name = df[df['host_name'].isnull()]

#display the output
missing_host_name

# end #
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	r
846	6104732	Rental unit in Toronto · 1 bedroom · 1 bed · 1...	31675651	NaN	NaN	Waterfront Communities-The Island	43.65095	-79.35694	
2919	17417181	Rental unit in Toronto · 1 bedroom · 2 beds · ...	75779190	NaN	NaN	Edenbridge-Humber Valley	43.68364	-79.51211	

Q3(c)

```
# your code start here #

#dropna removes rows where there are missing vals
#subset parameter specifies the 'host_name' col
df = df.dropna(subset=['host_name'])

#shape returns a tuple, where the first element is the row and the second is the col
#index 0 returns the number of col
new_row = df.shape[0]

#print the number of col
print('the new row count is:', new_row)

#calculate new percentages vals
new_missing_percentages = df.isnull().mean().sort_values(ascending=False)

#new line for better visuals
print('\n')

#displaying the output
new_missing_percentages
# end #
```

```
the new row count is: 20384
```

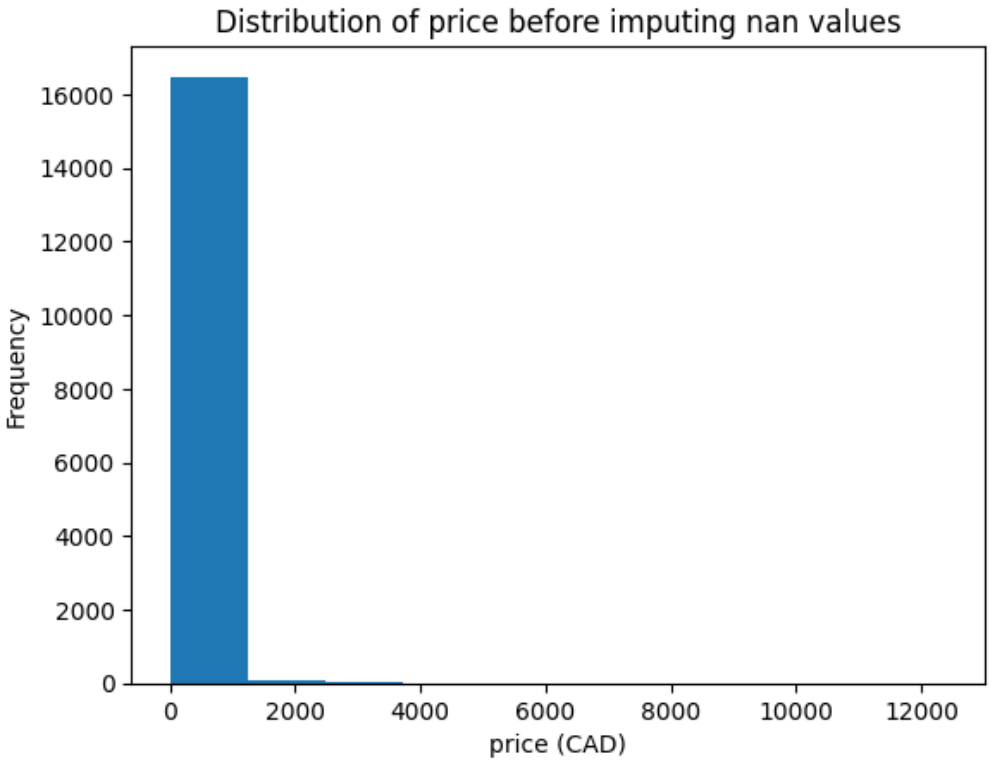
```
neighbourhood_group               1.000000
license                           0.536107
rating                            0.404582
last_review                       0.257898
reviews_per_month                 0.257898
price                             0.185881
latitude                          0.000000
longitude                         0.000000
room_type                         0.000000
name                              0.000000
minimum_nights                    0.000000
number_of_reviews                 0.000000
neighbourhood                     0.000000
host_name                         0.000000
calculated_host_listings_count    0.000000
availability_365                  0.000000
number_of_reviews_ltm             0.000000
host_id                           0.000000
id                                0.000000
dtype: float64
```

Q3(d)

```
# Do NOT modify this code cell

# summary statistics of price before imputing nan values
df['price'].describe()
```

```
count    16595.000000
mean       177.039168
std        333.108977
min          8.000000
25%         75.000000
50%        120.000000
75%        195.000000
max      12400.000000
Name: price, dtype: float64
```

```
# Do NOT modify this code cell
plt.hist(df['price'])
plt.xlabel('price (CAD)')
plt.ylabel('Frequency')
plt.title("Distribution of price before imputing nan values")
plt.show()
```




```
print(f"Before imputing null values, 'price' column has: {df['price'].isnull().sum()} null values")


# your code start here #

#.sum() calculates the count of NaN before imputation
missing_val_prices_pre_imputation = df['price'].isnull().sum()

#prints the output
print(df['price'])

#groupby groups the neighbourhood col data
#.transform('mean') returns a series of data consistent with the original df
avg_price_by_neighborhood = df.groupby('neighbourhood')['price'].transform('mean')

#fills the missing vals with the mean values calculated above
df['price'] = df['price'].fillna(avg_price_by_neighborhood)

#to verify the missing vals have been filled in; counts after impuatation
missing_val_prices_post_imputation = df['price'].isnull().sum()

#prints the output
print('\n')
print(df['price'])

# end #

print(f"After imputing null values, 'price' column now has: {df['price'].isnull().sum()} null values")
```

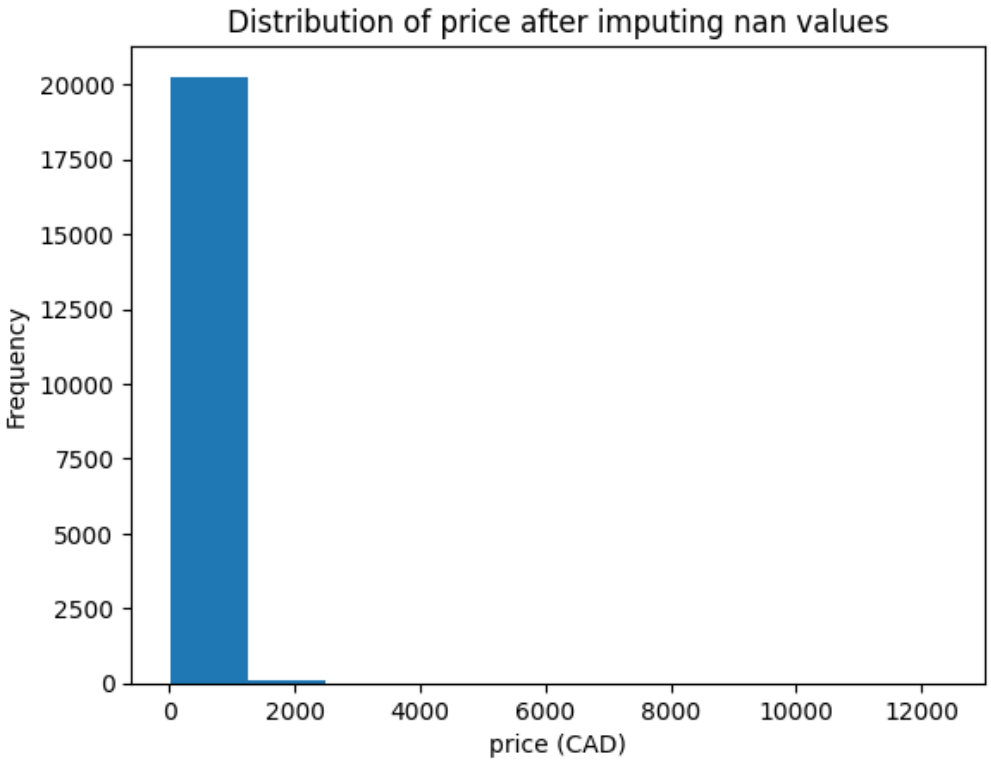
```
Before imputing null values, 'price' column has: 3789 null values
0          NaN
1        100.0
2        145.0
3         75.0
```

```
4        134.0
         ...
20381    139.0
20382     56.0
20383    169.0
20384    252.0
20385     70.0
Name: price, Length: 20384, dtype: float64


0        170.971326
1        100.000000
2        145.000000
3         75.000000
4        134.000000
            ...
20381    139.000000
20382     56.000000
20383    169.000000
20384    252.000000
20385     70.000000
Name: price, Length: 20384, dtype: float64
After imputing null values, 'price' column now has: 0 null values
```

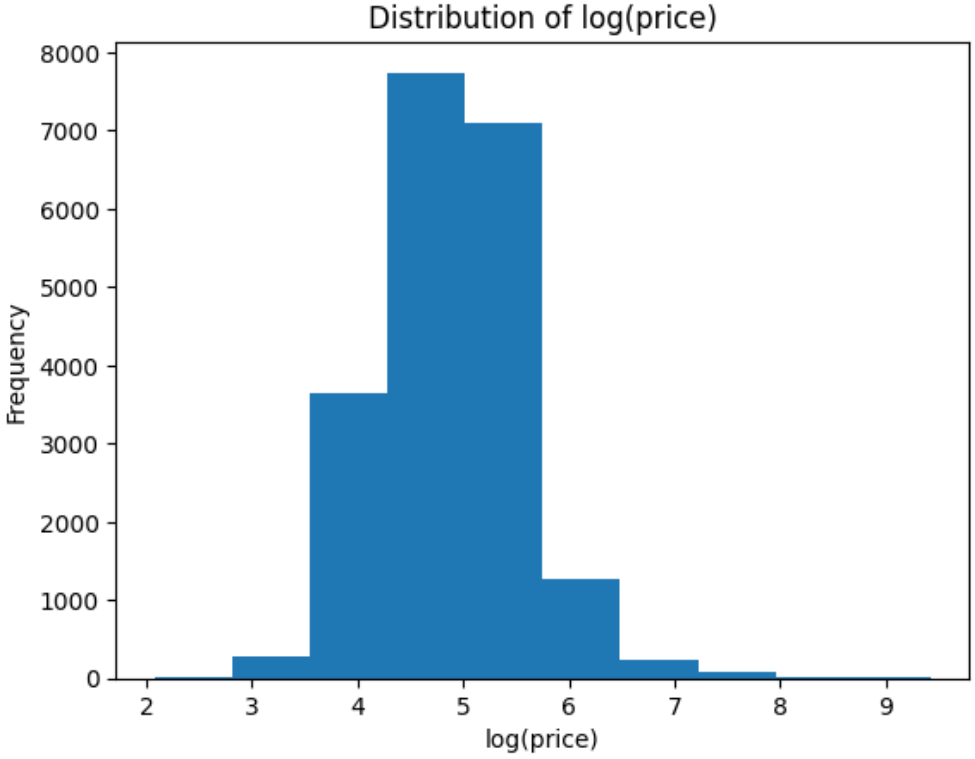
```
# Do NOT modify this code cell

# summary statistics of price after imputing nan values
df['price'].describe()
```

```
count    20384.000000
mean       178.209381
std        301.435569
min          8.000000
25%         84.000000
50%        135.000000
75%        205.548649
max      12400.000000
Name: price, dtype: float64
```

```
# Do NOT modify this code cell
plt.hist(df['price'])
plt.xlabel('price (CAD)')
plt.ylabel('Frequency')
plt.title("Distribution of price after imputing nan values")
plt.show()
```




```
# Do NOT modify this code cell

plt.hist(np.log(df['price']))
plt.xlabel('log(price)')
plt.ylabel('Frequency')
plt.title("Distribution of log(price)")
plt.show()
```



Q4(a)

```
# your code start here #

#calculates the z-score of the log of the price col
df['z-score_of_log_price'] = (np.log(df['price']) - np.log(df['price']).mean()) / np.log(df['price']).std()

#display the output
print(df['z-score_of_log_price'])
# end #
```

```
0             NaN
1       -0.295936
2        0.208631
3       -0.686596
4        0.101496
           ...
20381    0.151244
20382   -1.083304
20383    0.416622
20384    0.959166
20385   -0.780285
Name: z-score_of_log_price, Length: 20386, dtype: float64
```

Q4(b)

```
cols_to_output = ["id", "name", "host_id", "host_name", "neighbourhood", "room_type", "price", "

zscore_threshold = 3

# your code start here #

#for the specified output col
output_col = ["id", "name", "host_id", "host_name", "neighbourhood", "room_type", "price", "numb

#identifies the top-priced listings
top_priced_listings = df[df['z-score_of_log_price'] > zscore_threshold]

#sorting the data using price
top_priced_listings = top_priced_listings[output_col].sort_values(by='price', ascending=False)
# end #

print(f"The shape of top_priced_listings is: {top_priced_listings.shape}")

top_priced_listings # Outputs top_priced_listings.
                    # Note, Google Colab may truncate this outputs to the first five and last fi
```

istings is: (132, 9)

id	name	host_id	host_name	neighbourhood	room_type	price	number_of_reviews	rating
83	Rental unit in Toronto · ★4.86 · 1 bedroom · 2...	12931053	Colin	Palmerston-Little Italy	Entire home/apt	12400.0	14	4.86
37	Boutique hotel in Toronto · ★4.67 · 1 bedroom ...	271838768	Bond	Church-Yonge Corridor	Private room	10000.0	3	4.67
96	Condo in Toronto · ★4.87 · Studio · 2 beds · 1...	206884960	Jenny	Waterfront Communities-The Island	Entire home/apt	10000.0	193	4.87
83	Condo in Toronto · ★4.58 · 1 bedroom · 2 beds ...	495792127	Luca	Waterfront Communities-The Island	Entire home/apt	10000.0	26	4.58
11	Rental unit in Toronto · ★4.68 · Studio · 1 be...	496441567	Jacob Willow James	Palmerston-Little Italy	Entire home/apt	10000.0	25	4.68
...	...	...	...	...	...	...	...	...
47	Home in Toronto · ★4.67 · 4 bedrooms · 5 beds ...	59038458	Chi Yin Andy	Moss Park	Entire home/apt	1155.0	3	4.67
11	Home in Toronto · ★5.0 · 4 bedrooms · 4 beds ·...	470071465	Xiaozhen	St.Andrew-Windfields	Entire home/apt	1150.0	10	5.0
07	Townhouse in Toronto · 4 bedrooms · 4 beds · 4...	325867297	Reservations	Palmerston-Little Italy	Entire home/apt	1143.0	0	NaN
29	Home in Toronto · ★New · 4 bedrooms · 4 beds ·...	51626541	Niki	Bedford Park-Nortown	Entire home/apt	1143.0	0	NaN
46	Home in Toronto · 2 bedrooms · 3 beds · 2 baths	384337721	Sophie U	University	Entire home/apt	1142.0	1	NaN

Q4(c)

```
# your code start here #

#identifies the bottom priced listings
bottom_priced_listings = df[df['z-score_of_log_price'] < -zscore_threshold]
# end #

print(f"The shape of bottom_priced_listings is: {bottom_priced_listings.shape}")

bottom_priced_listings # outputs bottom_priced_listings
```