In [1]:

```
import matplotlib.pyplot as plt
from tabulate import tabulate
import matplotlib as mat
import seaborn as sns
import pandas as pd
import numpy as np
```

In [2]:

```
print("Pandas version: ",pd.__version__)
print("Seaborn version: ",sns.__version__)
print("Matplotlib version: ",mat.__version__)
```

```
Pandas version:  1.3.4
Seaborn version:  0.11.2
Matplotlib version:  3.4.3
```

In [3]:

```
#By using this we can get the data what we are using .
#Here df is the DataFrame
```

In [7]:

```
df1 = pd.read_csv('data (1) (1).csv')
```

In [8]:

```
print(df1)
```

```
10688      acept  not furnished      3100             15000
10689  not acept      furnished       980              6000
10690      acept      furnished      1585             12000
10691      acept  not furnished         0              1400

       property tax (R$)  fire insurance (R$)  total (R$)
0                    211                   42        5618
1                   1750                   63        7973
2                      0                   41        3841
3                     22                   17        1421
4                     25                   11         836
...                  ...                  ...         ...
10687                 24                   22        1926
10688                973                  191       19260
10689                332                   78        7390
10690                279                  155       14020
10691                165                   22        1587

[10692 rows x 13 columns]
```

In [9]:

```
df1.describe()
```

Out[9]:

	area	rooms	bathroom	parking spaces	floor	hoa (R$)
count	10692.000000	10692.000000	10692.000000	10692.000000	10692.000000	1.069200e+04
mean	149.217920	2.506079	2.236813	1.609147	5.067995	1.174022e+03
std	537.016942	1.171266	1.407198	1.589521	6.069050	1.559231e+04
min	11.000000	1.000000	1.000000	0.000000	0.000000	0.000000e+00
25%	56.000000	2.000000	1.000000	0.000000	1.000000	1.700000e+02
50%	90.000000	2.000000	2.000000	1.000000	3.000000	5.600000e+02
75%	182.000000	3.000000	3.000000	2.000000	8.000000	1.237500e+03
max	46335.000000	13.000000	10.000000	12.000000	301.000000	1.117000e+06

In [10]:

```
df1.head()   #First n rows
```

Out[10]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	property tax (R$)	ir
0	São Paulo	70	2	1	1	7	acept	furnished	2065	3300	211	
1	São Paulo	320	4	4	0	20	acept	not furnished	1200	4960	1750	
2	Porto Alegre	80	1	1	1	6	acept	not furnished	1000	2800	0	
3	Porto Alegre	51	2	1	0	2	acept	not furnished	270	1112	22	
4	São Paulo	25	1	1	0	1	not acept	not furnished	0	800	25	

In [11]:

```
df1.info()     #
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10692 entries, 0 to 10691
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   city                 10692 non-null  object
 1   area                 10692 non-null  int64 
 2   rooms                10692 non-null  int64 
 3   bathroom             10692 non-null  int64 
 4   parking spaces       10692 non-null  int64 
 5   floor                10692 non-null  int64 
 6   animal               10692 non-null  object
 7   furniture            10692 non-null  object
 8   hoa (R$)             10692 non-null  int64 
 9   rent amount (R$)     10692 non-null  int64 
 10  property tax (R$)    10692 non-null  int64 
 11  fire insurance (R$)  10692 non-null  int64 
 12  total (R$)           10692 non-null  int64 
dtypes: int64(10), object(3)
memory usage: 1.1+ MB
```

In [12]:

```
df1.tail()      #displays the last five rows of the dataframe by default
```

Out[12]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	proper tax (R
10687	Porto Alegre	63	2	1	1	5	not acept	furnished	402	1478	
10688	São Paulo	285	4	4	4	17	acept	not furnished	3100	15000	9
10689	Rio de Janeiro	70	3	3	0	8	not acept	furnished	980	6000	3
10690	Rio de Janeiro	120	2	2	2	8	acept	furnished	1585	12000	2
10691	São Paulo	80	2	1	0	0	acept	not furnished	0	1400	1

LEVEL 0 ANALYSIS (FROM HERE GETTING START)

In [ ]:

```
#So now we have to Read the first 5 rows , what the data is given for us.
```

In [13]:

```
df1.head(5) # head prints the top 5 rows
```

Out[13]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	property tax (R$)	ir
0	São Paulo	70	2	1	1	7	acept	furnished	2065	3300	211	
1	São Paulo	320	4	4	0	20	acept	not furnished	1200	4960	1750	
2	Porto Alegre	80	1	1	1	6	acept	not furnished	1000	2800	0	
3	Porto Alegre	51	2	1	0	2	acept	not furnished	270	1112	22	
4	São Paulo	25	1	1	0	1	not acept	not furnished	0	800	25	

```
#Now by using the shape part we can get that how many rows and columns are there.
```

In [14]:

```
print(df1.shape)      #Here the shape defines that there are how many number of rows  and c
```

```
(10692, 13)
```

In [ ]:

```
#Now we are going to display all the columns names
```

In [15]:

```
print(df1.columns) #So by getting it we can see that totally we have 13 columns
```

```
Index(['city', 'area', 'rooms', 'bathroom', 'parking spaces', 'floor',
       'animal', 'furniture', 'hoa (R$)', 'rent amount (R$)',
       'property tax (R$)', 'fire insurance (R$)', 'total (R$)'],
      dtype='object')
```

In [ ]:

```
#Now one step left to check is there any null values columns are there or not.
```

In [16]:

```
df1.isnull().sum()
```

Out[16]:

```
city                  0
area                  0
rooms                 0
bathroom              0
parking spaces        0
floor                 0
animal                0
furniture             0
hoa (R$)              0
rent amount (R$)      0
property tax (R$)     0
fire insurance (R$)   0
total (R$)            0
dtype: int64
```

In [17]:

```
print(df1.size)
```

```
138996
```

In [ ]:

```
#LEVEL 1 ANALYSIS(STARTS FROM HERE)
```

```
#Firslty the work is that we have to find out that the data given to us is categorical or
numerical after
#that we can tak ethe decisions .
```

```
#Seperating the CAT OR NUM VARIABLES:
#CAT IS CATEGORIAL AND NUM IS NUMERICAL
```

In [18]:

```
def seprate_data_types(df1):
    categorical = []
    continuous = []
    for column in df1.columns:                 # looping on the number of columns
        if df1[column].nunique() < 50:        # nunique:  returns unique values in the colum
                                                    #here 50 is the hyperparamter and can b
            categorical.append(column)
        else:
            continuous.append(column)

    return categorical, continuous


categorical, continuous = seprate_data_types(df1)
# Calling the function


from tabulate import tabulate
table = [categorical, continuous]
print(tabulate({"Categorical":categorical,
                "continuous": continuous}, headers = ["categorical", "continuous"]))
```

```
categorical     continuous
--------------  -------------------
city            area
rooms           hoa (R$)
bathroom        rent amount (R$)
parking spaces  property tax (R$)
floor           fire insurance (R$)
animal          total (R$)
furniture
```

```
#After doing some of the code we have get which columns are cat and what are continuous.
```

```
#LEVEL 1 ANALYTICS(UNIVARIATE ANALYSIS)
```

In [ ]:

```
Here in this we have to go for the define a fucntion information .
```

In [19]:

```
def info_of_cat(col):       #Information about the columns
    print(f"Unique values in {col} are: {df1[col].unique()}")
    # unique: returns the unique values in the col

    print(f"Mode of {col} is {df1[col].mode()[0]}")
    # mode: returns the mode of the column

    print(f"Number of missing values in {col} is {df1[col].isnull().sum()}")
    # isnull().sum() counts the number of null in dataframe

    if df1[col].isnull().sum() > 0:  #` check if null values are present
        print(f"\nThere are null values in the {col} column")
```

```
#LEVEL 1 ANALYSIS OF CATEGORIAL DATA
```

Q1-ANALYSIS THE CITY COLUMN AND CHECK WHICH CITY HAS MORE COUNTING.

In [20]:

```
info_of_cat("city")
```

```
Unique values in city are: ['São Paulo' 'Porto Alegre' 'Rio de Janeiro' 'Cam
pinas' 'Belo Horizonte']
Mode of city is São Paulo
Number of missing values in city is 0
```

In [ ]:

```
So here we get the missing value as 0 ,and here we get some of the unique city values are w
```

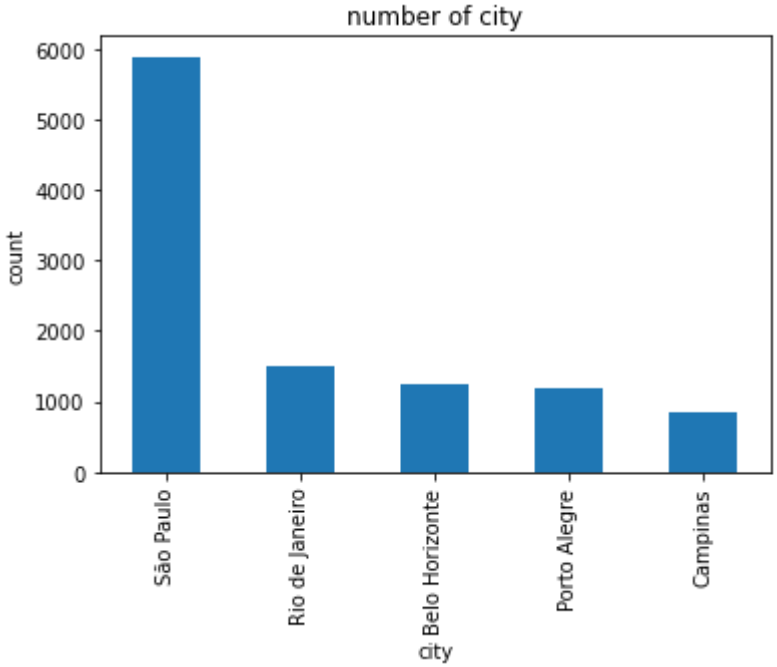
In [21]:

```
city_mode = df1.city.mode()[0]
print(city_mode)
df1["city"].fillna(city_mode, inplace = True)
```

```
São Paulo
```

In [22]:

```
df1['city'].value_counts().plot(kind='bar')
plt.title('number of city')
plt.xlabel('city')
plt.ylabel('count')
plt.show()
```




```
CITY IS NOMINAL DATATYPE
```

```
#Intrepetations - The visualization of city names are there in the Excel sheet.Now the
city have the largest count is the Sao Paulo that is which is in higgest peak and rest
are at the average part.
```

In [ ]:

```
#So now for more understanding the thinks we are going to plot the graph then we can see th
```

```
LEVEL 1 Analysis of Categorical Data (Univariate Analysis)
```

Q2) Define that how would you shwo that which city having the biggets number of bathroom,also plot the graph to explain it

in details?

In [23]:

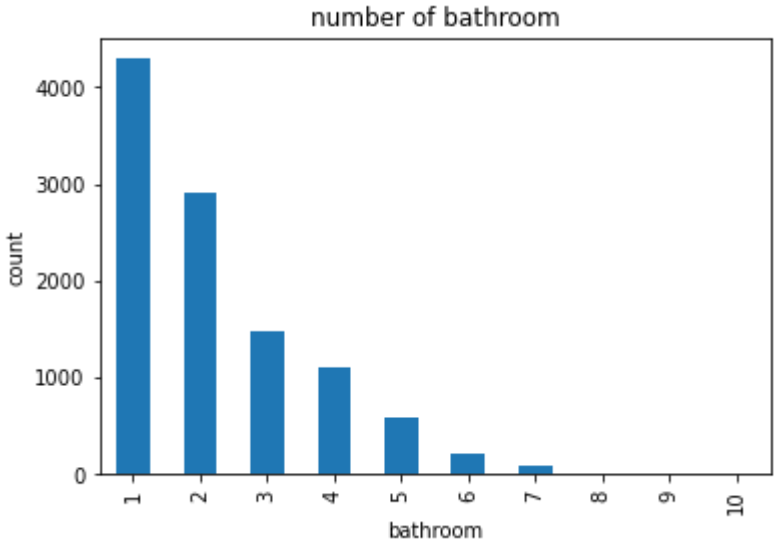
```
info_of_cat("bathroom")
```

```
Unique values in bathroom are: [ 1  4  3  2  6  5  7  9  8 10]
Mode of bathroom is 1
Number of missing values in bathroom is 0
```

```
#So by evaluating the information of categorial data of rooms ,what i have found that the
mode of rooms are 3
#and there is not a single value.
```

In [24]:

```
df1['bathroom'].value_counts().plot(kind='bar')
plt.title('number of bathroom')
plt.xlabel('bathroom')
plt.ylabel('count')
plt.show()
```



```
#Interpretation:Sao Paulo having the largest number of bathroom as companred to the other
city bathroom.
```

Outliers Analysis and Treatment:

In [25]:

```
df1.describe()
```

Out[25]:

	area	rooms	bathroom	parking spaces	floor	hoa (R$)	r
count	10692.000000	10692.000000	10692.000000	10692.000000	10692.000000	1.069200e+04	10
mean	149.217920	2.506079	2.236813	1.609147	5.067995	1.174022e+03	3
std	537.016942	1.171266	1.407198	1.589521	6.069050	1.559231e+04	3
min	11.000000	1.000000	1.000000	0.000000	0.000000	0.000000e+00	
25%	56.000000	2.000000	1.000000	0.000000	1.000000	1.700000e+02	1
50%	90.000000	2.000000	2.000000	1.000000	3.000000	5.600000e+02	2
75%	182.000000	3.000000	3.000000	2.000000	8.000000	1.237500e+03	5
max	46335.000000	13.000000	10.000000	12.000000	301.000000	1.117000e+06	45

In [ ]:

```
Interpretation: Area are  usually have 10  dependents and some are occupied /
```

4. Analysis of furniture Level: furniture of a Housing

In [26]:

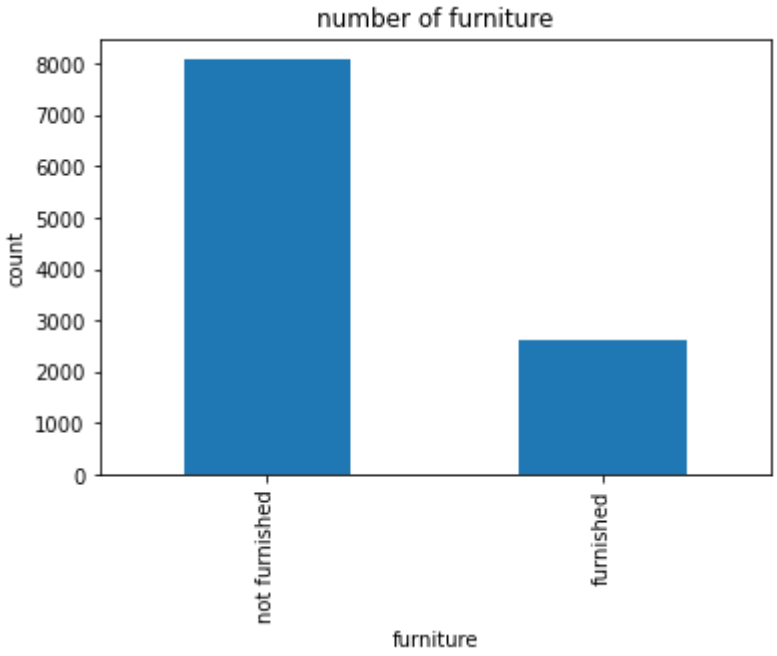
```
info_of_cat( "furniture")
```

```
Unique values in furniture are: ['furnished' 'not furnished']
Mode of furniture is not furnished
Number of missing values in furniture is 0
```

```
#So here in the furniture category we have totally two parts that are furnished
#and not furnished,and there is not a single null values.
```

In [27]:

```
df1['furniture'].value_counts().plot(kind='bar')
plt.title('number of furniture')
plt.xlabel('furniture')
plt.ylabel('count')
plt.show()
```




```
#So now from graph we have observed that the Not furnished is more than the furnished.
```

5. Analysis of Floor: Floor status

In [28]:

```
info_of_cat( "floor")
```

```
Unique values in floor are: [  7  20   6   2   1   0   4   3  10  11  24   9
8  17  18   5  13  15
  16  14  26  12  21  19  22  27  23  35  25  46  28  29 301  51  32]
Mode of floor is 0
Number of missing values in floor is 0
```

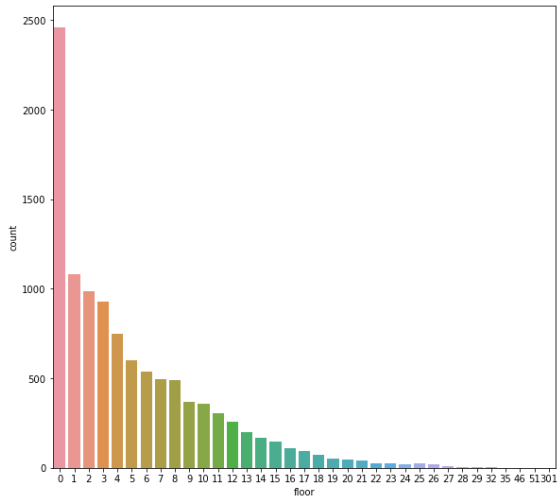
Q3) How would you analysis the graph of the floor and also show that which floor having the biggest

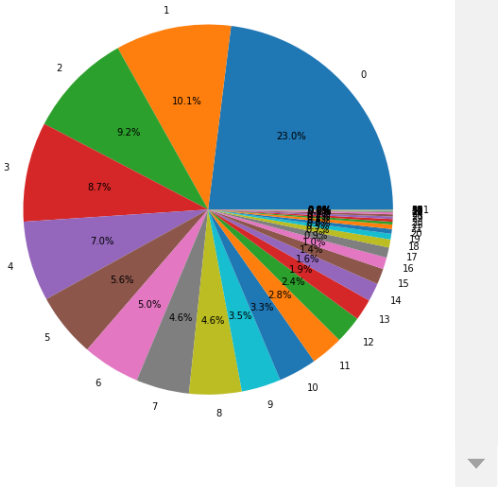
number?

In [29]:

```
fig, ax = plt.subplots(1, 2, figsize = (21,9))
percentage = (df1["floor"].value_counts()/len(df1))*100      # value count is a function to
                                                              #keeping its classe
print(percentage)
sns.countplot(x = df1["floor"], ax = ax[0])
labels = list(df1["floor"].value_counts().index)            # value_counts returns the coun
                                                               # is accessed by t
ax[1].pie(percentage,labels = labels, autopct= "%0.1f%%")          # autpct: is the way ho
plt.show()
```

```
0      23.017209
1      10.110363
2       9.212495
3       8.707445
4       6.995885
5       5.611672
6       5.041152
7       4.648335
8       4.582866
9       3.451178
10      3.338945
11      2.833895
12      2.403666
13      1.870557
14      1.589974
15      1.374860
16      1.019454
17      0.897868
18      0.701459
19      0.495698
20      0.411523
21      0.392817
25      0.233820
23      0.233820
22      0.224467
26      0.187056
24      0.177703
27      0.074822
28      0.056117
29      0.046764
32      0.018706
35      0.009353
46      0.009353
301     0.009353
51      0.009353
Name: floor, dtype: float64
```






```
#So after observing from the graph we cans ee that the grould floor is the higgest then
the any other floor
#As there are 23% are the ground floor.
```

6. Analysis of Income Category: Income categories of card holders

In [30]:

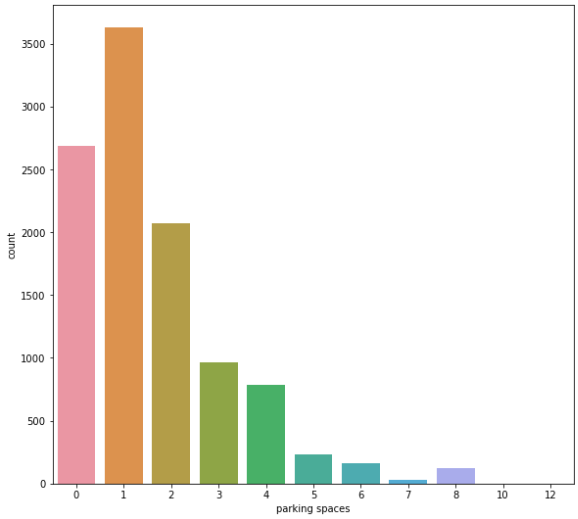
```
info_of_cat( "parking spaces")
```

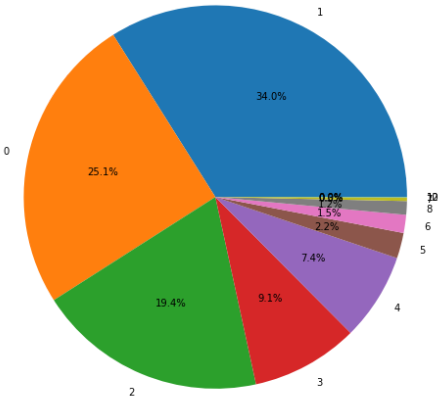
```
Unique values in parking spaces are: [ 1  0  7  4  2  6  3  8  5 10 12]
Mode of parking spaces is 1
Number of missing values in parking spaces is 0
```

In [31]:

```
fig, ax = plt.subplots(1, 2, figsize = (21,9))
percentage = (df1["parking spaces"].value_counts()/len(df1))*100      # value count is a fu
                                                                        #keeping its classe
print(percentage)
sns.countplot(x = df1["parking spaces"], ax = ax[0])
labels = list(df1["parking spaces"].value_counts().index)            # value_counts returns
                                                                        # is accessed by t
ax[1].pie(percentage,labels = labels, autopct= "%0.1f%%")           # autpct: is the way ho
plt.show()
```

```
1     33.950617
0     25.093528
2     19.360269
3      9.053498
4      7.379349
5      2.151141
6      1.524504
8      1.150393
7      0.308642
10     0.018706
12     0.009353
Name: parking spaces, dtype: float64
```





In [ ]:

```
#As of observing it we can get to know  that the parking spaces is bit higgeser in 1 in sao
```

LEVEL 1 Analysis of Numerical Columns

In [33]:

```
def info_of_numerical(col):
    print(f"The mean of the {col} is {df1[col].mean()}")
    print(f"The median of the {col} is {df1[col].median()}")
    print(f"The mode of the {col} is {df1[col].mode()[0]}")
    print(f"The standard deviation of the {col} is {df1[col].std()}")
    print(f"Number of missing values in the {col} is {df1[col].isnull().sum()}")
```

In [ ]:

```
#Now here we can find out some of the user defined functions,for calculating mean, median,
#and the count of all null values.
```

Q4) Analyse the data and find out the user defined fuction and elaborate it using the graph?

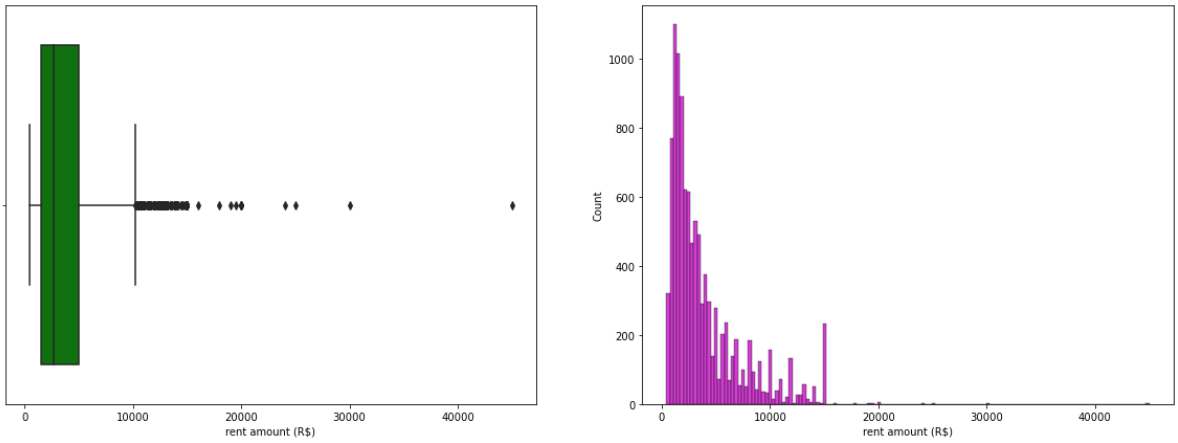
In [36]:

```
info_of_numerical("rent amount (R$)")
```

```
The mean of the rent amount (R$) is 3896.247194163861
The median of the rent amount (R$) is 2661.0
The mode of the rent amount (R$) is 2500
The standard deviation of the rent amount (R$) is 3408.5455176710816
Number of missing values in the rent amount (R$) is 0
```

In [38]:

```
fig, ax = plt.subplots(1, 2, figsize= (20, 7))                # Creating the space for the 2
sns.histplot(x = df1["rent amount (R$)"], ax =ax[1], color = "m") # ax[1] means image will
sns.boxplot(x = df1["rent amount (R$)"], ax = ax[0], color = "g") # ax[0] means image will
plt.show()
```



In this case, the rent amount follows a normal distribution. There are rent amount with its approx as 40 k.
The analytics team decides to replace the rent amount greater than 40k with the average rent amount.

Outliers Analysis and Treatment:

In [ ]:

```
#Q5) Show the value of mean ,median mode and std deviation ,and plot the graph?
```

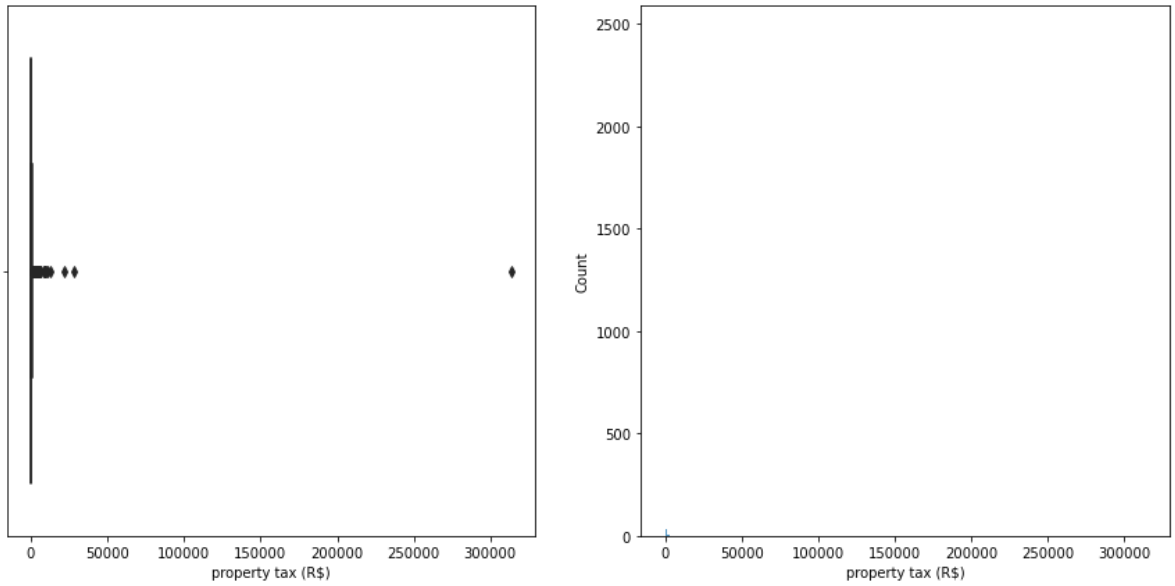
In [43]:

```
info_of_numerical("property tax (R$)")
```

```
The mean of the property tax (R$) is 366.70435839880287
The median of the property tax (R$) is 125.0
The mode of the property tax (R$) is 0
The standard deviation of the property tax (R$) is 3107.832321161917
Number of missing values in the property tax (R$) is 0
```

In [50]:

```
fig, ax = plt.subplots(1, 2, figsize= (15, 7))
sns.histplot(x = df1['property tax (R$)'])                          # x axis contains the re
                                                                     # y axis has the count
sns.boxplot(x = df1['property tax (R$)'], ax = ax[0], color = "g") # c : color: b is black
plt.show()
```



In [ ]:

```
#Q6) Difference between the rent amount and the fire insurance  at the higgest balance on t
```

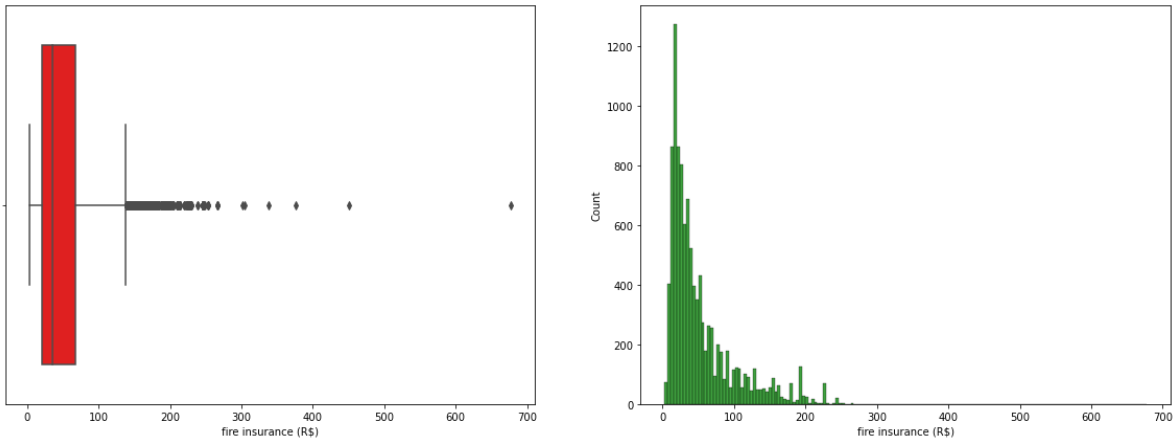
In [51]:

```
info_of_numerical("fire insurance (R$)")
```

```
The mean of the fire insurance (R$) is 53.300879161990274
The median of the fire insurance (R$) is 36.0
The mode of the fire insurance (R$) is 16
The standard deviation of the fire insurance (R$) is 47.768030930197206
Number of missing values in the fire insurance (R$) is 0
```

In [52]:

```
fig, ax = plt.subplots(1, 2, figsize= (20, 7))
sns.histplot(df1["fire insurance (R$)"], ax = ax[1], color= "g")
sns.boxplot(x = df1['fire insurance (R$)'], ax = ax[0], color = "r")
plt.show()
```



In [ ]:

```
So by observing the analysis we can find that the right skewed data has the more umber of s
is the data for this.
```

In [ ]:

```
#Q7)  Analysis of the total (R$): Number of transactions made for the hosuing data
```

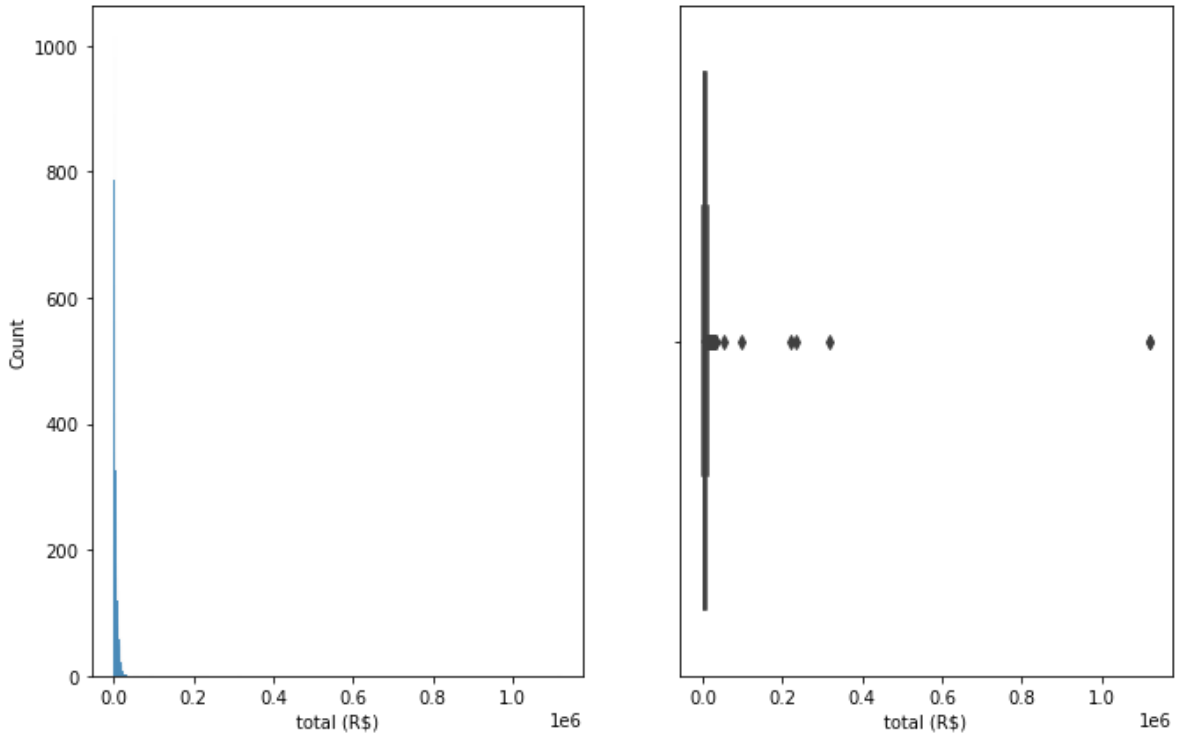
In [53]:

```
info_of_numerical("total (R$)")
```

```
The mean of the total (R$) is 5490.4869996258885
The median of the total (R$) is 3581.5
The mode of the total (R$) is 2555
The standard deviation of the total (R$) is 16484.72591235027
Number of missing values in the total (R$) is 0
```

In [57]:

```
fig, ax = plt.subplots(1, 2, figsize= (11, 7))
sns.histplot(x = df1['total (R$)'], ax = ax[0]) #ax[0]: graph(Histogram) will be ploted at
sns.boxplot(x = df1["total (R$)"], ax = ax[1])  #ax[1]: graph(Boxplot) will be ploted at 0t
plt.show()
```



In [ ]:

```
Now there are some of the utilization is that and we want to get the total of overall housi
```

In [ ]:

```
#Q8 ) Analysis of the Total(R$): It is a ratio of Total Revolving Balance for the Housing d
```

In [65]:

```
info_of_numerical("total (R$)")
```

```
The mean of the total (R$) is 5490.4869996258885
The median of the total (R$) is 3581.5
The mode of the total (R$) is 2555
The standard deviation of the total (R$) is 16484.72591235027
Number of missing values in the total (R$) is 0
```

In [73]:

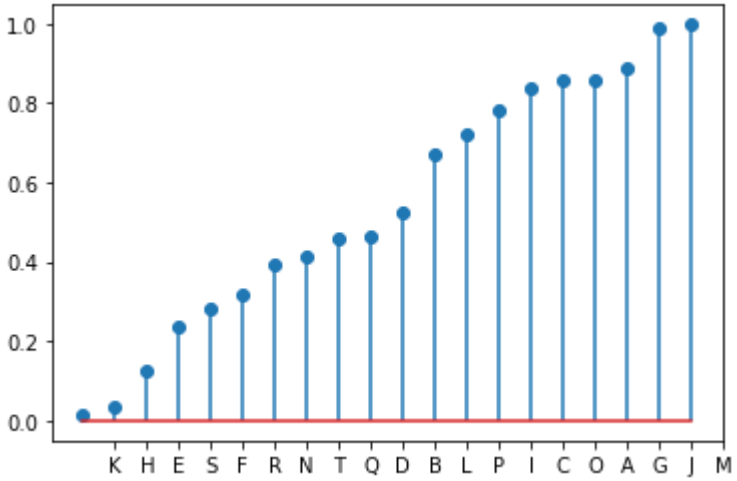
```
df = pd.DataFrame({'group':list(map(chr, range(65, 85))), 'values':np.random.uniform(size=2

# Reorder it following the values:
housing_df = df.sort_values(by='values')
my_range=range(1,len(df1.index)+1)

# Make the plot
plt.stem(housing_df['values'])
plt.xticks( my_range, housing_df['group'])
```

Out[73]:

```
([<matplotlib.axis.XTick at 0x15a7dde6c10>,
  <matplotlib.axis.XTick at 0x15a7dde6be0>,
  <matplotlib.axis.XTick at 0x15a72c4b220>,
  <matplotlib.axis.XTick at 0x15a7de52460>,
  <matplotlib.axis.XTick at 0x15a7de52bb0>,
  <matplotlib.axis.XTick at 0x15a7de5d340>,
  <matplotlib.axis.XTick at 0x15a7de5da90>,
  <matplotlib.axis.XTick at 0x15a7de5a220>,
  <matplotlib.axis.XTick at 0x15a7de5a970>,
  <matplotlib.axis.XTick at 0x15a7de5d4f0>,
  <matplotlib.axis.XTick at 0x15a7de52400>,
  <matplotlib.axis.XTick at 0x15a7de5af10>,
  <matplotlib.axis.XTick at 0x15a7de656a0>,
  <matplotlib.axis.XTick at 0x15a7de65df0>,
  <matplotlib.axis.XTick at 0x15a7de5ff70>,
  <matplotlib.axis.XTick at 0x15a7de65ca0>,
  <matplotlib.axis.XTick at 0x15a7de5a670>,
  <matplotlib.axis.XTick at 0x15a7de5f940>,
  <matplotlib.axis.XTick at 0x15a7de6d220>,
  <matplotlib.axis.XTick at 0x15a7de6d970>],
 [Text(1, 0, 'K'),
  Text(2, 0, 'H'),
  Text(3, 0, 'E'),
  Text(4, 0, 'S'),
  Text(5, 0, 'F'),
  Text(6, 0, 'R'),
  Text(7, 0, 'N'),
  Text(8, 0, 'T'),
  Text(9, 0, 'Q'),
  Text(10, 0, 'D'),
  Text(11, 0, 'B'),
  Text(12, 0, 'L'),
  Text(13, 0, 'P'),
  Text(14, 0, 'I'),
  Text(15, 0, 'C'),
  Text(16, 0, 'O'),
  Text(17, 0, 'A'),
  Text(18, 0, 'G'),
  Text(19, 0, 'J'),
  Text(20, 0, 'M')])
```



In [ ]:

```
#Interpretations: - By observing this we can see that the Groups are incrasing .
```

Level 3 Further Analysis Questions

```
#Q9 ) Which city having the larger churn?
```

In [75]:

```
df1 = pd.read_csv('data (1) (1).csv')
```

In [76]:

```
df1
```

Out[76]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	proper tax (R
0	São Paulo	70	2	1	1	7	acept	furnished	2065	3300	2
1	São Paulo	320	4	4	0	20	acept	not furnished	1200	4960	17
2	Porto Alegre	80	1	1	1	6	acept	not furnished	1000	2800	
3	Porto Alegre	51	2	1	0	2	acept	not furnished	270	1112	
4	São Paulo	25	1	1	0	1	not acept	not furnished	0	800	
...	...	...	...	...	...	...	...	...	...	...	
10687	Porto Alegre	63	2	1	1	5	not acept	furnished	402	1478	
10688	São Paulo	285	4	4	4	17	acept	not furnished	3100	15000	9
10689	Rio de Janeiro	70	3	3	0	8	not acept	furnished	980	6000	3
10690	Rio de Janeiro	120	2	2	2	8	acept	furnished	1585	12000	2
10691	São Paulo	80	2	1	0	0	acept	not furnished	0	1400	1

10692 rows × 13 columns

In [77]:

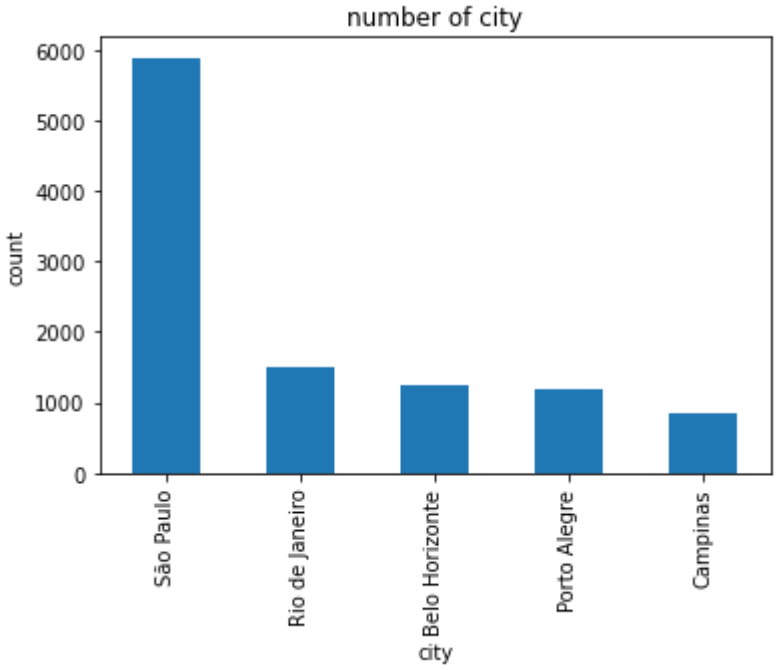
```
df1.head()
```

Out[77]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	property tax (R$)	ir
0	São Paulo	70	2	1	1	7	acept	furnished	2065	3300	211	
1	São Paulo	320	4	4	0	20	acept	not furnished	1200	4960	1750	
2	Porto Alegre	80	1	1	1	6	acept	not furnished	1000	2800	0	
3	Porto Alegre	51	2	1	0	2	acept	not furnished	270	1112	22	
4	São Paulo	25	1	1	0	1	not acept	not furnished	0	800	25	

In [81]:

```
df1['city'].value_counts().plot(kind='bar')
plt.title('number of city')
plt.xlabel('city')
plt.ylabel('count')
plt.show()
```

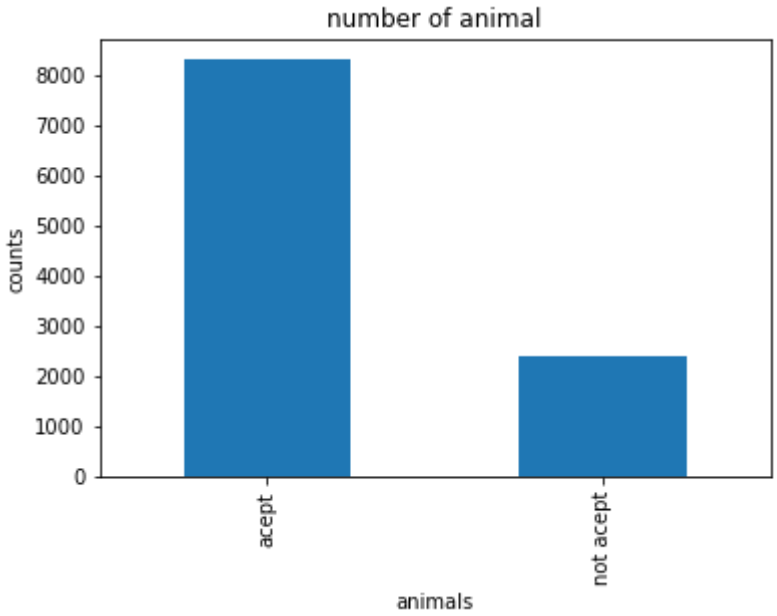


In [ ]:

```
#Q 10) Analyze the churn of Animals that will be allowed or not.
```

In [87]:

```
df1['animal'].value_counts().plot(kind='bar')
plt.title('number of animal')
plt.xlabel('animals')
plt.ylabel('counts')
plt.show()
```




```
#So by observing this we can say that there is high amount of animals are accept to keep
for anumals in some the housing and
#some are not allowed.
```

In [ ]:

```
# Q 11) City-Wise analysis of the housing who are  done the highest curn in both the scenar
```

In [ ]:

```
data = df1[(df1["city"] == 2) | (df1["rooms"] == 3)]
g = sns.FacetGrid(data, col = "total (R$)", height = 7)
g.map(sns.histplot, "city")
plt.show()
```

```
#Final Analysis: The final observation is that some of the places like sao paulo is best
for the housing and it is nearer to some of the
#city to work from Home.
```

--------- THE END ----------