In [26]:
```
punctuations = '''!()-{}[];:"'\,<>./?@#$%^&*_~`'''
```

In [27]:
```
my_str = "Hello!!!, he said ...and went."
```

In [31]:
```
no_punct =""
```

In [34]:
```
for char in my_str:
 if(char not in punctuations):
    no_punct = no_punct + char
    print(no_punct)
```

```
H
He
Hel
Hell
Hello
Hello
Hello h
Hello he
Hello he
Hello he s
Hello he sa
Hello he sai
Hello he said
Hello he said
Hello he said a
Hello he said an
Hello he said and
Hello he said and
Hello he said and w
Hello he said and we
Hello he said and wen
Hello he said and went
```

In [35]:
```
import re
s = "Hello!!!, he said ...and went."
s = re.sub(r'[^\w\s]','',s)
# not of word and space character
print(s)
```

```
Hello he said and went
```

In [40]:
```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize
example_text = "Hello Mr. Pravin, how are you doing today? The weather in lona
print(sent_tokenize(example_text))
print(word_tokenize(example_text))
```

```
['Hello Mr. Pravin, how are you doing today?', 'The weather in lonavala israi
ny.', 'The sky is full of cloud.']
['Hello', 'Mr.', 'Pravin', ',', 'how', 'are', 'you', 'doing', 'today', '?',
'The', 'weather', 'in', 'lonavala', 'israiny', '.', 'The', 'sky', 'is', 'ful
l', 'of', 'cloud', '.']

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Pranali\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [41]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
e", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'i
t', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but',
'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'wit
h', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'af
ter', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when',
'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most',
'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'th
an', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'shoul
d', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren',
"aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'might
n', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'sh
ouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'w
ouldn', "wouldn't"]

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Pranali\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [43]:
```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()
input_str = "There are several types of stemming algorithms."
input_str = nltk.word_tokenize(input_str)
print (input_str)
for word in input_str:
 print(stemmer.stem(word))
```

```
['There', 'are', 'several', 'types', 'of', 'stemming', 'algorithms', '.']
there
are
sever
type
of
stem
algorithm
.
```

In [45]:
```
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
lemmatizer = WordNetLemmatizer()
input_str = "There are several cities with mice."
input_str = nltk.word_tokenize(input_str)
print (input_str)
for word in input_str:
 print(lemmatizer.lemmatize(word))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Pranali\AppData\Roaming\nltk_data...

['There', 'are', 'several', 'cities', 'with', 'mice', '.']
There
are
several
city
with
mouse
.
```

In [50]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("C:\\Users\\Pranali\\Downloads\\records.csv")
df.head()
print(df)

df.info()
```

```
   ID    Name    Role  Salary
0   1  Pankaj  Editor   10000
1   2    Lisa  Editor    8000
2   3   David  Author    6000
3   4     Ram  Author    4000
4   5  Anupam  Author    5000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      5 non-null      int64
 1   Name    5 non-null      object
 2   Role    5 non-null      object
 3   Salary  5 non-null      int64
dtypes: int64(2), object(2)
memory usage: 292.0+ bytes
```

In [51]:
```
updated_df = df.dropna(axis=1)
print(updated_df)
updated_df.info()
```

```
   ID    Name    Role  Salary
0   1  Pankaj  Editor   10000
1   2    Lisa  Editor    8000
2   3   David  Author    6000
3   4     Ram  Author    4000
4   5  Anupam  Author    5000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      5 non-null      int64
 1   Name    5 non-null      object
 2   Role    5 non-null      object
 3   Salary  5 non-null      int64
dtypes: int64(2), object(2)
memory usage: 292.0+ bytes
```

In [52]:
```
updated_df = df.dropna(axis=0)
print(updated_df)
updated_df.info()
```

```
   ID    Name    Role  Salary
0   1  Pankaj  Editor   10000
1   2    Lisa  Editor    8000
2   3   David  Author    6000
3   4     Ram  Author    4000
4   5  Anupam  Author    5000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      5 non-null      int64
 1   Name    5 non-null      object
 2   Role    5 non-null      object
 3   Salary  5 non-null      int64
dtypes: int64(2), object(2)
memory usage: 292.0+ bytes
```

In [53]:
```
updated_df = df
updated_df['Salary']=updated_df['Salary'].fillna(updated_df['Salary'].mean())
print(updated_df)
```

```
   ID    Name    Role  Salary
0   1  Pankaj  Editor   10000
1   2    Lisa  Editor    8000
2   3   David  Author    6000
3   4     Ram  Author    4000
4   5  Anupam  Author    5000
```

In [54]:
```
updated_df = df
updated_df['Salaryismissing'] = updated_df['Salary'].isnull()
print(updated_df)
```

```
   ID    Name    Role  Salary  Salaryismissing
0   1  Pankaj  Editor   10000            False
1   2    Lisa  Editor    8000            False
2   3   David  Author    6000            False
3   4     Ram  Author    4000            False
4   5  Anupam  Author    5000            False
```

In [55]:
```
testdf = df[df['Salary'].isnull()==True]
traindf = df[df['Salary'].isnull()==False]
traindf.drop("Salary",axis=1,inplace=True)
testdf.drop("Salary",axis=1,inplace=True)
print(traindf)
```

```
   ID    Name    Role  Salaryismissing
0   1  Pankaj  Editor            False
1   2    Lisa  Editor            False
2   3   David  Author            False
3   4     Ram  Author            False
4   5  Anupam  Author            False

C:\Users\Pranali\AppData\Local\Temp\ipykernel_24728\201493639.py:4: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  testdf.drop("Salary",axis=1,inplace=True)
```

In [56]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [57]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [58]:
```
data=pd.read_csv("C:\\Users\\Pranali\\Downloads\\house.csv")
```

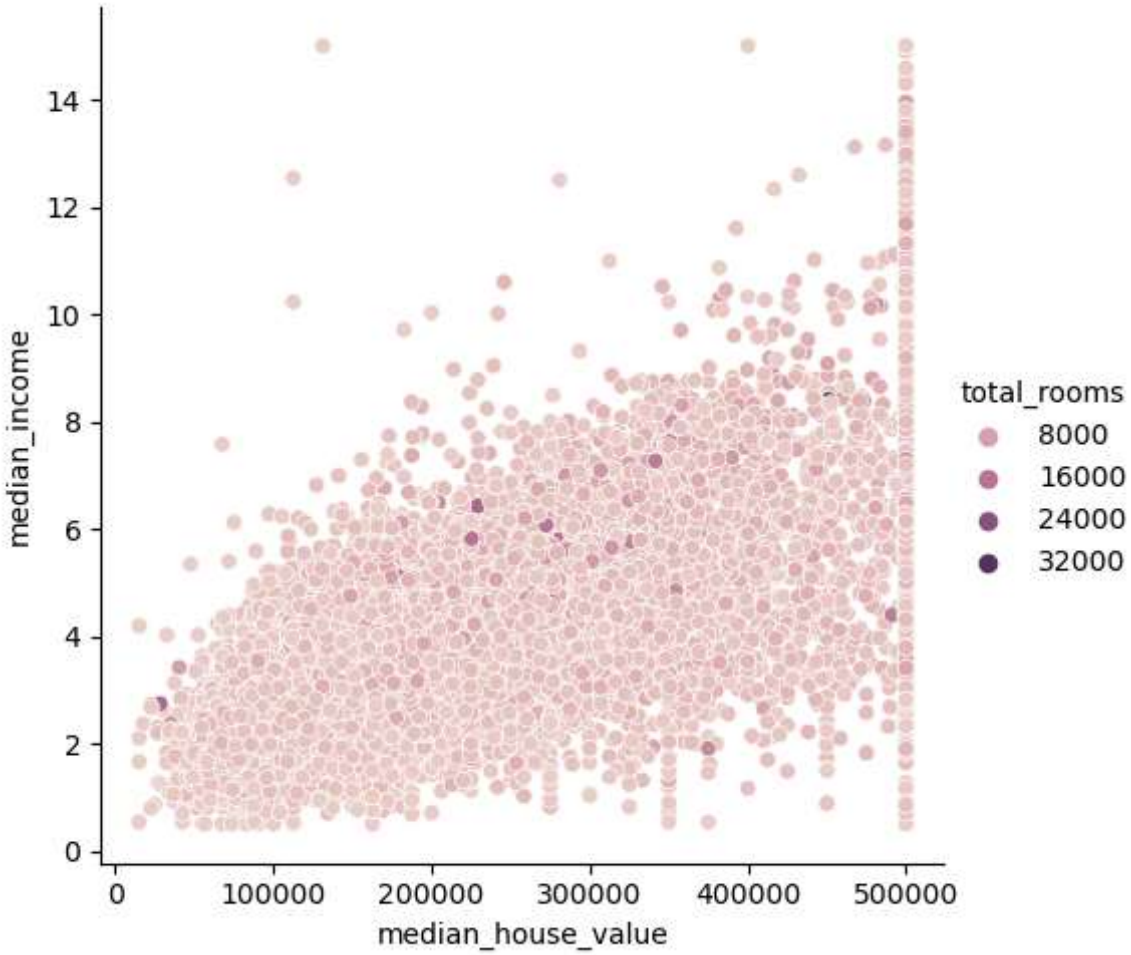
In [59]:

```
print(data.columns)
print(data.head(10))
print(data.describe())

print(data.shape)
print(data.isnull().sum())
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -114.31     34.19                  15         5612            1283
1    -114.47     34.40                  19         7650            1901
2    -114.56     33.69                  17          720             174
3    -114.57     33.64                  14         1501             337
4    -114.57     33.57                  20         1454             326
5    -114.58     33.63                  29         1387             236
6    -114.58     33.61                  25         2907             680
7    -114.59     34.83                  41          812             168
8    -114.59     33.61                  34         4789            1175
9    -114.60     34.83                  46         1497             309

   population  households  median_income  median_house_value
0        1015         472         1.4936               66900
1        1129         463         1.8200               80100
2         333         117         1.6509               85700
3         515         226         3.1917               73400
4         624         262         1.9250               65500
5         671         239         3.3438               74000
6        1841         633         2.6768               82400
7         375         158         1.7083               48500
8        3134        1056         2.1782               58400
9         787         271         2.1908               48100
          longitude      latitude  housing_median_age   total_rooms  \
count  17000.000000  17000.000000        17000.000000  17000.000000
mean    -119.562108     35.625225           28.589353   2643.664412
std        2.005166      2.137340           12.586937   2179.947071
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.790000     33.930000           18.000000   1462.000000
50%     -118.490000     34.250000           29.000000   2127.000000
75%     -118.000000     37.720000           37.000000   3151.250000
max     -114.310000     41.950000           52.000000  37937.000000

       total_bedrooms    population    households  median_income  \
count    17000.000000  17000.000000  17000.000000   17000.000000
mean       539.410824   1429.573941    501.221941       3.883578
std        421.499452   1147.852959    384.520841       1.908157
min          1.000000      3.000000      1.000000       0.499900
25%        297.000000    790.000000    282.000000       2.566375
50%        434.000000   1167.000000    409.000000       3.544600
75%        648.250000   1721.000000    605.250000       4.767000
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        17000.000000
mean        207300.912353
std         115983.764387
min          14999.000000
25%         119400.000000
50%         180400.000000
75%         265000.000000
max         500001.000000
(17000, 9)
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

In [60]:
```
sns.relplot(x='median_house_value', y= 'median_income', hue= 'total_rooms', da
plt.show()
```



In [61]:
```
train =data.drop(['median_house_value','longitude','latitude','housing_median_
test =data['median_house_value']
x_train, x_test, y_train, y_test = train_test_split(train, test, test_size=0.3
```

In [62]:
```
regr= LinearRegression()
regr.fit(x_train, y_train)
pred = regr.predict(x_test)
print(pred)
print(regr.score(x_test, y_test))
```

```
[153579.02594907 221485.91195112  91641.96769418 ... 184433.07649541
 224004.49656511 185189.89991136]
0.541520003241859
```

In [ ]: