Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

Import libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# To import dataset
df=pd.read_csv('21_cities.csv')
df
```

Out[2]:

	id	name	state_id	state_code	state_name	country_id	country_code	country
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afgh
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afgh
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afgh
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afgh
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afgh
...	...	...	...	...	...	...	...	
150449	131496	Redcliff	1957	MI	Midlands Province	247	ZW	Zin
150450	131502	Shangani	1957	MI	Midlands Province	247	ZW	Zin
150451	131503	Shurugwi	1957	MI	Midlands Province	247	ZW	Zin
150452	131504	Shurugwi District	1957	MI	Midlands Province	247	ZW	Zin
150453	131508	Zvishavane District	1957	MI	Midlands Province	247	ZW	Zin

150454 rows × 11 columns

In [3]:
```
# To display top 10 rows
df.head(10)
```

Out[3]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_name
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanistan
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanistan
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanistan
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanistan
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanistan
5	131	Wākhān	3901	BDS	Badakhshan	1	AF	Afghanistan
6	72	Ghormach	3871	BDG	Badghis	1	AF	Afghanistan
7	108	Qala i Naw	3871	BDG	Badghis	1	AF	Afghanistan
8	54	Baghlān	3875	BGL	Baghlan	1	AF	Afghanistan
9	140	Ḩukūmatī Dahanah-ye Ghōrī	3875	BGL	Baghlan	1	AF	Afghanistan

Data Cleaning and Pre-Processing

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150454 entries, 0 to 150453
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            150454 non-null  int64  
 1   name          150454 non-null  object 
 2   state_id      150454 non-null  int64  
 3   state_code    150129 non-null  object 
 4   state_name    150454 non-null  object 
 5   country_id    150454 non-null  int64  
 6   country_code  150406 non-null  object 
 7   country_name  150454 non-null  object 
 8   latitude      150454 non-null  float64
 9   longitude     150454 non-null  float64
 10  wikiDataId    147198 non-null  object 
dtypes: float64(2), int64(3), object(6)
memory usage: 12.6+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	id	state_id	country_id	latitude	longitude
count	150454.000000	150454.000000	150454.000000	150454.000000	150454.000000
mean	76407.091689	2678.377677	140.658460	31.556175	2.369557
std	44357.755335	1363.513591	70.666123	22.813220	68.012770
min	1.000000	1.000000	1.000000	-75.000000	-179.121980
25%	38160.250000	1451.000000	82.000000	19.000000	-58.468150
50%	75975.500000	2174.000000	142.000000	40.684720	8.669980
75%	115204.750000	3905.000000	207.000000	47.239220	27.750000
max	153528.000000	5116.000000	247.000000	73.508190	179.466000

In [6]:
```
df.columns
```

Out[6]:
```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

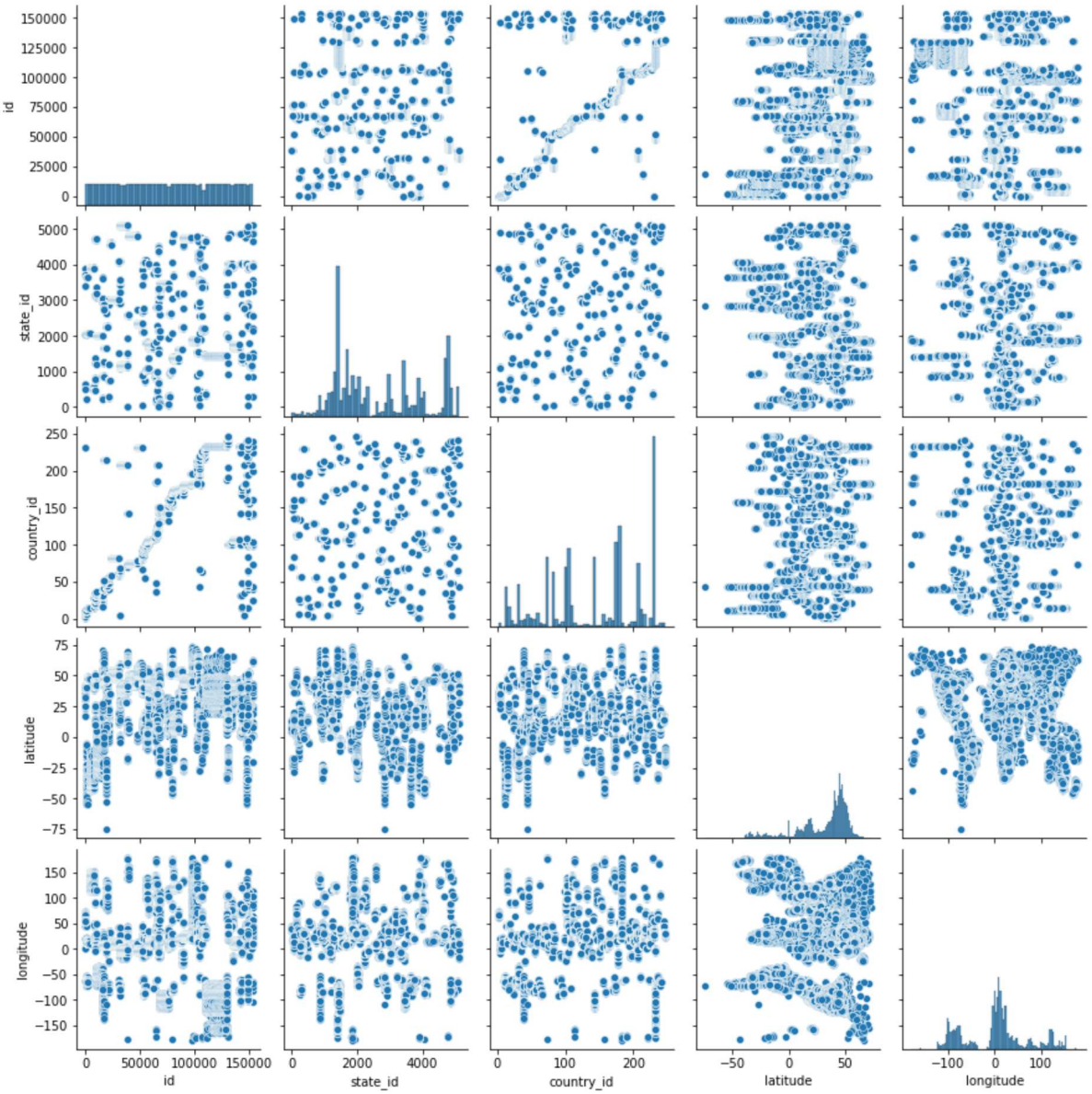
In [7]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[7]:
```
Index(['id', 'name', 'state_id', 'state_name', 'country_id', 'country_name',
       'latitude', 'longitude'],
      dtype='object')
```

EDA and Visualization

In [8]:
```
sns.pairplot(a)
```

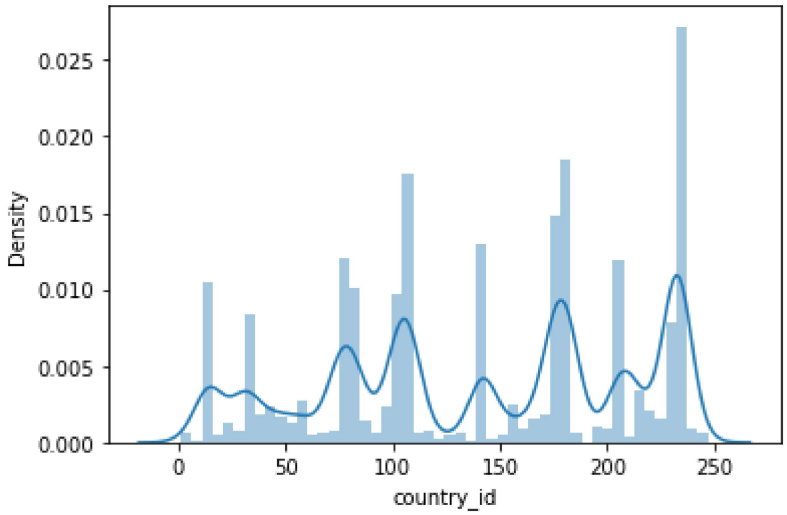
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x263f72ab8b0>
```



In [9]:
```
sns.distplot(a['country_id'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

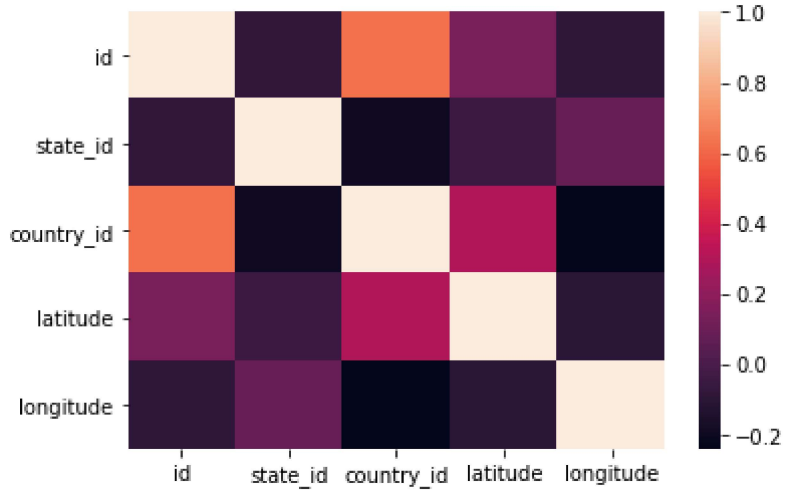
Out[9]: <AxesSubplot:xlabel='country_id', ylabel='Density'>



In [10]:
```
a1=a[['id', 'state_id', 'country_id',
       'latitude', 'longitude']]
```

In [11]:
```
sns.heatmap(a1.corr())
```

Out[11]: <AxesSubplot:>



To Train the Model - Model Building

We are going to train Linear Regression model:We need to split out data into two variables x

In [12]:
```
x=a1[['id', 'state_id',
       'latitude', 'longitude']]
y=a1['country_id']
```

To split my dataset into training and test data

In [13]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [14]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
68.67627247067769
```

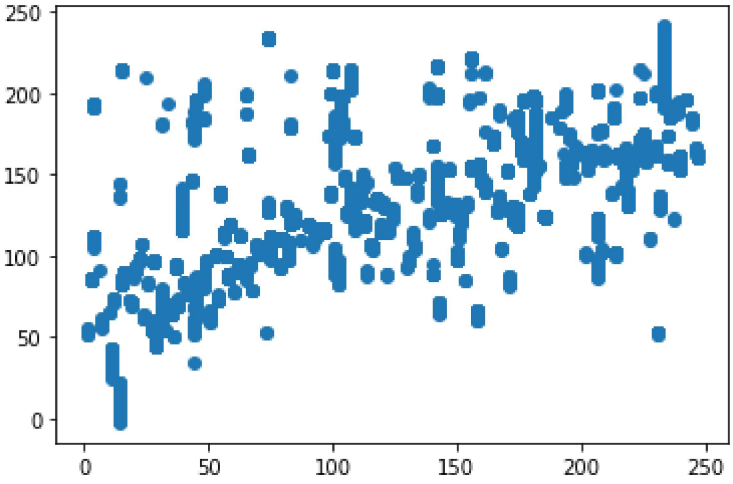
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
id	0.000924
state_id	-0.006384
latitude	0.598729
longitude	-0.164477

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x263fa28cf40>



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.4843963386141138
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]: Ridge(alpha=10)

In [21]:
```

rr.score(x_train,y_train)
```

Out[21]: 0.4842877623680948

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]: 0.48439633850484554

In [23]:
```
rr.score(x_test,y_test)
```

Out[23]: 0.48439633850484554

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]: Lasso(alpha=10)

In [25]:
```
la.score(x_test,y_test)
```

Out[25]: 0.48433491223991165

In [26]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[26]:
```
ElasticNet()
```

In [27]:
```
print(en.coef_)
```

```
[ 0.00092427 -0.00638511  0.59718135 -0.1643968 ]
```

In [28]:
```
print(en.intercept_)
```

```
68.71797524717141
```

In [29]:
```
print(en.predict(x_test))
```

```
[132.43744748 126.11758008  72.64365246 ... 103.20199564 220.63517466
 143.03276914]
```

In [30]:
```
print(en.score(x_test,y_test))
```

```
0.48439452985988607
```

Evaluation Metrics

In [31]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pre
```

```
Mean Absolytre Error: 39.655732544318354
Mean Squared Error: 2568.034488204265
Root Mean Squared Error: 50.6757781213497
```

In [32]:
```
import pickle
```

In [34]:
```
filename='prediction3'
pickle.dump(lr,open(filename,'wb'))
```

In [ ]: