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('20_states.csv')
df
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007
...	...	...	...	...	...	...	...	...
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201

5077 rows × 9 columns

In [3]:

```
# To display top 10 rows
df.head(10)
```

Out[3]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	lon
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.8
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.7
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.7
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.8
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.8
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495	66.0
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328	62.2
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561	64.9
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059	68.4
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578	64.9

Data Cleaning and Pre-Processing

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5077 entries, 0 to 5076
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            5077 non-null   int64  
 1   name          5077 non-null   object 
 2   country_id    5077 non-null   int64  
 3   country_code  5063 non-null   object 
 4   country_name  5077 non-null   object 
 5   state_code    5072 non-null   object 
 6   type          1597 non-null   object 
 7   latitude      5008 non-null   float64
 8   longitude     5008 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 357.1+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	id	country_id	latitude	longitude
count	5077.000000	5077.000000	5008.000000	5008.000000
mean	2609.765413	133.467599	27.576415	17.178713
std	1503.376799	72.341160	22.208161	61.269334
min	1.000000	1.000000	-54.805400	-178.116500
25%	1324.000000	74.000000	11.399747	-3.943859
50%	2617.000000	132.000000	34.226432	17.501792
75%	3905.000000	201.000000	45.802822	41.919647
max	5220.000000	248.000000	77.874972	179.852222

In [6]:

```
df.columns
```

Out[6]:

```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [7]:

```
a = df.dropna(axis='columns')
a.columns
```

Out[7]:

```
Index(['id', 'name', 'country_id', 'country_name'], dtype='object')
```

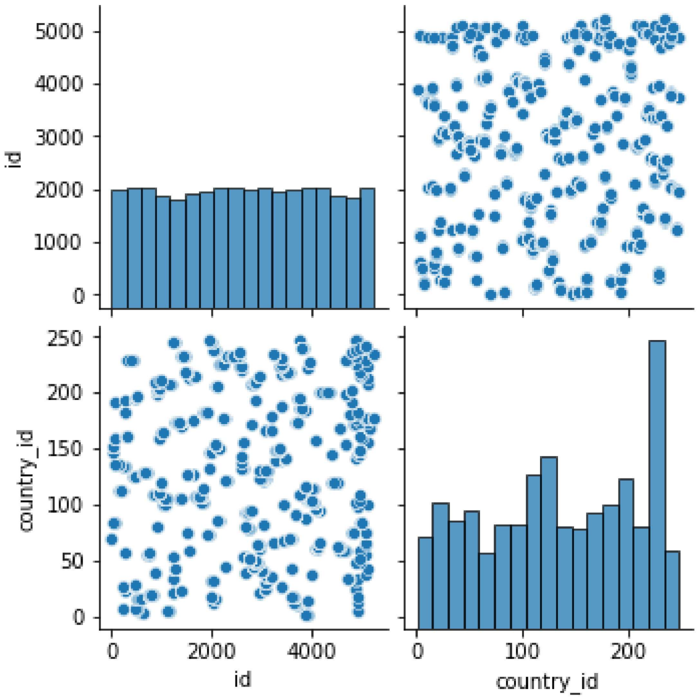
EDA and Visualization

In [8]:

```
sns.pairplot(a)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x22239f86af0>
```



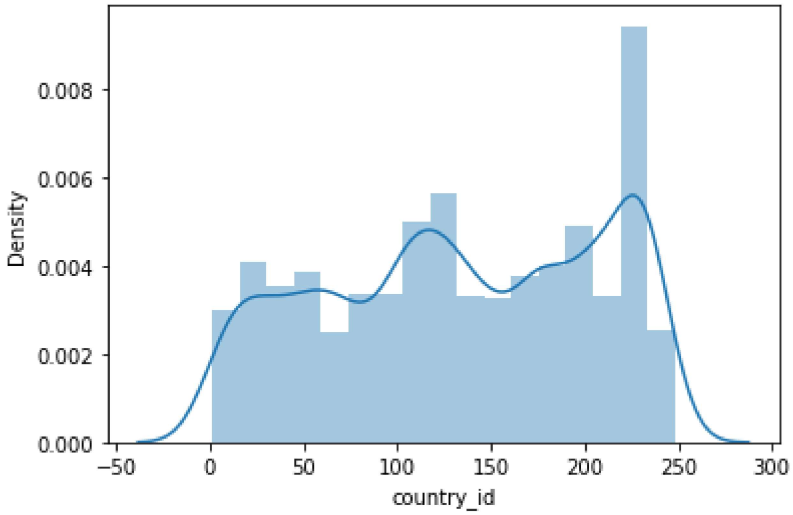
In [9]:

```
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]:

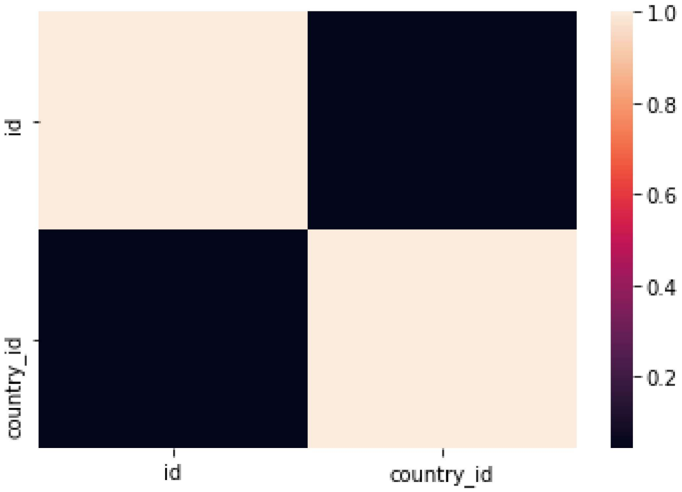
```
<AxesSubplot:xlabel='country_id', ylabel='Density'>
```



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

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 and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [12]:
```
x=a1[['id']]
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_)
```

```
127.57310018216988
```

In [16]:

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

Out[16]:

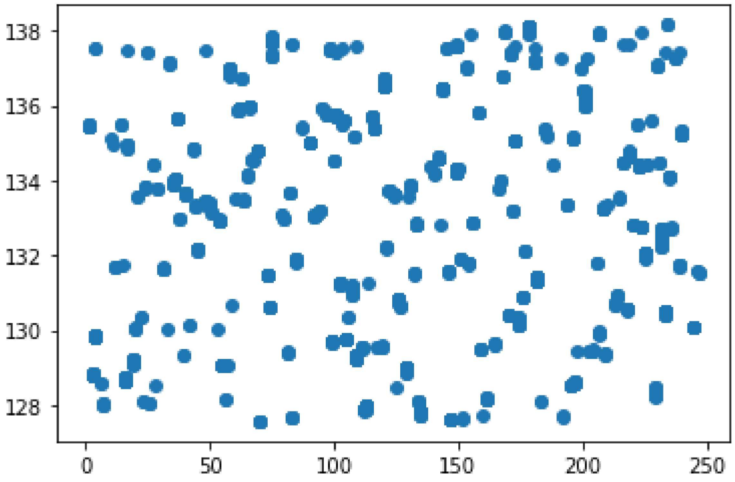
	Co-efficient
id	0.002029

In [17]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:

```
<matplotlib.collections.PathCollection at 0x2223b38a820>
```



In [18]:

```
print(lr.score(x_test,y_test))
```

```
0.0017083297723273771
```

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]:

```
0.0017342131031994334
```

In [22]:

```
rr.score(x_test,y_test)
```

Out[22]:

```
0.0017083297715219103
```

In [23]:

```
rr.score(x_test,y_test)
```

Out[23]:

```
0.0017083297715219103
```

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]:
```
0.001706910407451523
```

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

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

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

```
[0.0020287]
```

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

```
127.5736789283807
```

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

```
[137.43316486 137.48996848 130.72019388 ... 127.95507468 136.5466226
 134.85265743]
```

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

```
0.0017082590891673854
```

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: 59.86992832773104
Mean Squared Error: 4933.335203113695
Root Mean Squared Error: 70.23770499606101
```

In [32]:
```

import pickle
```

In [33]:
```
filename='prediction2'
pickle.dump(lr,open(filename,'wb'))
```

In [ ]: