Data Cleaning

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

In [3]:
```
df = pd.read_csv(r"C:\Users\user\Downloads\21_cities.csv")[0:500]
df
```

Out[3]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_nam
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanista
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanista
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanista
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanista
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanista
...	...	...	...	...	...	...	...	
495	31357	Ighram	1131	15	Tizi Ouzou	4	DZ	Alger
496	31371	L'Arbaa Naït Irathen	1131	15	Tizi Ouzou	4	DZ	Alger
497	31380	Mekla	1131	15	Tizi Ouzou	4	DZ	Alger
498	31451	Timizart	1131	15	Tizi Ouzou	4	DZ	Alger
499	31454	Tirmitine	1131	15	Tizi Ouzou	4	DZ	Alger

500 rows × 11 columns

In [4]:
```
# to display info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            500 non-null    int64
 1   name          500 non-null    object
 2   state_id      500 non-null    int64
 3   state_code    500 non-null    object
 4   state_name    500 non-null    object
 5   country_id    500 non-null    int64
 6   country_code  500 non-null    object
 7   country_name  500 non-null    object
 8   latitude      500 non-null    float64
 9   longitude     500 non-null    float64
 10  wikiDataId    500 non-null    object
dtypes: float64(2), int64(3), object(6)
memory usage: 43.1+ KB
```

In [5]:
```
# t display summerize the data
df.describe()
```

Out[5]:

	id	state_id	country_id	latitude	longitude
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	26692.318000	1803.800000	3.188000	36.670717	19.469789
std	38318.954881	1467.410462	1.122129	3.095059	23.689201
min	50.000000	609.000000	1.000000	22.785000	-8.147430
25%	183.750000	639.000000	3.000000	35.007045	3.285403
50%	31253.500000	1122.000000	4.000000	36.313130	6.891475
75%	31383.250000	3873.000000	4.000000	39.885672	20.223260
max	146224.000000	4902.000000	4.000000	42.367980	73.349280

In [6]:
```
# to display columes
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')
```

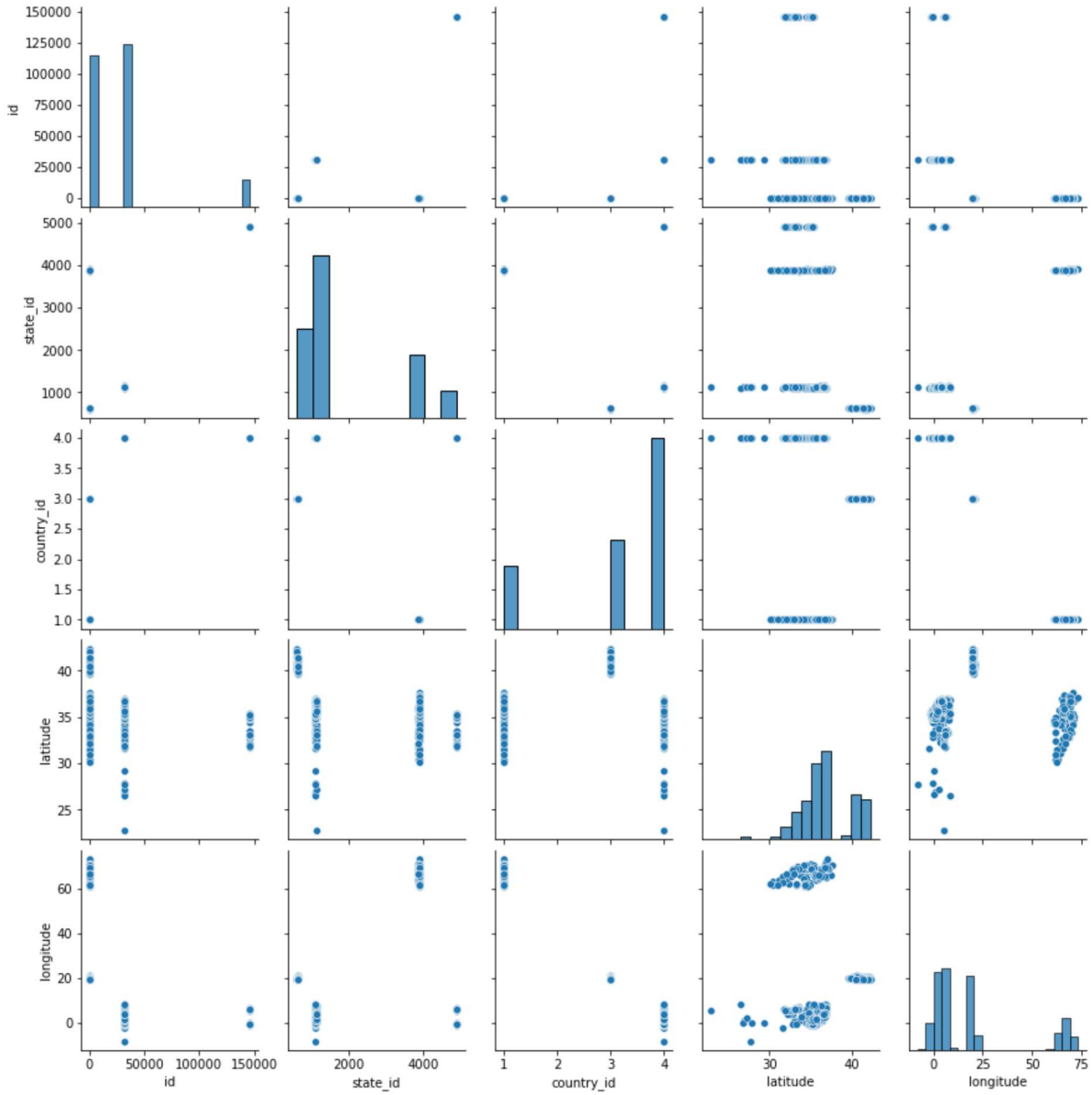
In [7]:
```
df.isna().sum()
```

Out[7]:
```
id              0
name            0
state_id        0
state_code      0
state_name      0
country_id      0
country_code    0
country_name    0
latitude        0
longitude       0
wikiDataId      0
dtype: int64
```

EDA and visualization

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

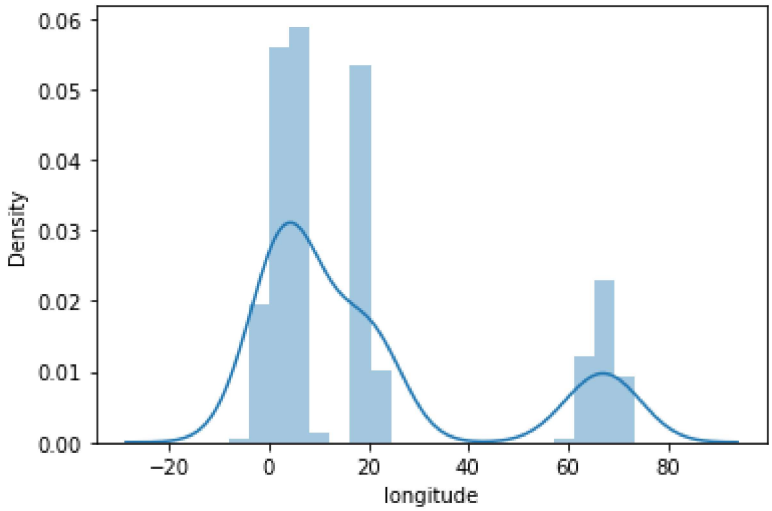
Out[8]: <seaborn.axisgrid.PairGrid at 0x21deb9e8f40>



In [9]:
```
# to display distribution graph for price column
sns.distplot(df['longitude'])
```

```
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)
```

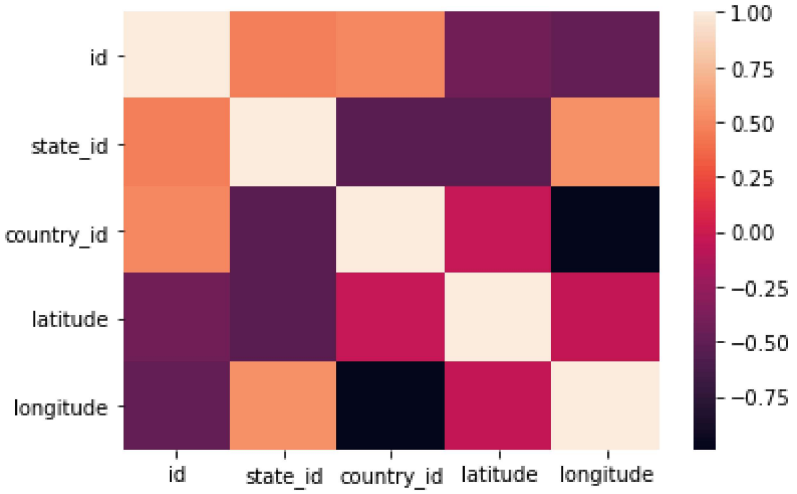
Out[9]: <AxesSubplot:xlabel='longitude', ylabel='Density'>



In [10]:
```
df1 = df[['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataId']]
```

In [11]:
```
# correlation map to find relationship
sns.heatmap(df1.corr())
```

Out[11]: <AxesSubplot:>



In [13]:
```
# Assign x and y for linear regression
x = df1[['id', 'state_id', 'country_id', 'latitude', 'longitude']]
y = df1['longitude']
```

In [14]:
```
# to split dataset into training data and test data

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 [15]:
```
#Linear Regression

from sklearn.linear_model import LinearRegression

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

Out[15]:
```
LinearRegression()
```

In [16]:
```
# intercept is value of c
print(lr.intercept_)
```

```
-3.552713678800501e-14
```

In [17]:
```
# co-efficient value of m
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

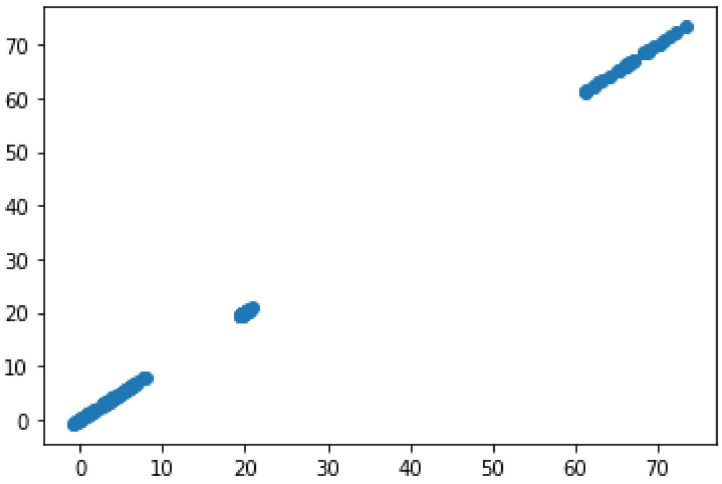
Out[17]:

	Co-efficient
id	-7.341984e-19
state_id	3.577685e-17
country_id	-2.767733e-15
latitude	3.655701e-17
longitude	1.000000e+00

In [18]:
```
#predict the graph in linear regression graph

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

Out[18]: <matplotlib.collections.PathCollection at 0x21df037b700>



In [19]:
```
#Accuracy of linear regression

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

```
1.0
```

In [20]:
```
lr.score(x_train,y_train)
```

Out[20]: 1.0

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[22]: 0.9999998015797438

In [23]:
```
rr.score(x_train,y_train)
```

Out[23]: 0.9999997959140918

In [24]:
```
lr = Lasso(alpha=10)
lr.fit(x_train,y_train)
lr.score(x_test,y_test)
```

Out[24]: 0.9855485281969368

In [25]:
```
lr.score(x_train,y_train)
```

Out[25]: 0.9859266245426145

Elastic

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

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

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

```
[-4.41253682e-05  1.20260713e-03 -0.00000000e+00  0.00000000e+00
  9.23469665e-01]
```

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

```
0.5045864566719906
```

In [29]:
```
print(es.score(x_test,y_test))
```

```
0.9998639481213156
```

Evaluation Model

In [30]:
```
from sklearn import metrics
```

In [31]:
```
print("Mean absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean absolute Error: 4.0690321482609685e-14
```

In [32]:
```
print("Mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean squared Error: 2.6298155091463318e-27
```

In [33]:
```
print("Root Mean squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean squared Error: 5.1281726854176154e-14
```

model saving

In [34]:
```
import pickle # pickle is used to model saving
```

In [35]:
```
filename ="21_cities prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [36]:
```
df.head(10)
```

Out[36]:

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

In [37]:
```
df.tail(10)
```

Out[37]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_name
490	31277	Boghni	1131	15	Tizi Ouzou	4	DZ	Algeria
491	31285	Boudjima	1131	15	Tizi Ouzou	4	DZ	Algeria
492	31303	Chemini	1131	15	Tizi Ouzou	4	DZ	Algeria
493	31321	Draa Ben Khedda	1131	15	Tizi Ouzou	4	DZ	Algeria
494	31345	Freha	1131	15	Tizi Ouzou	4	DZ	Algeria
495	31357	Ighram	1131	15	Tizi Ouzou	4	DZ	Algeria
496	31371	L'Arbaa Naït Irathen	1131	15	Tizi Ouzou	4	DZ	Algeria
497	31380	Mekla	1131	15	Tizi Ouzou	4	DZ	Algeria
498	31451	Timizart	1131	15	Tizi Ouzou	4	DZ	Algeria
499	31454	Tirmitine	1131	15	Tizi Ouzou	4	DZ	Algeria

In [ ]:
```

```