In [167]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from bs4 import BeautifulSoup as bs
import requests as re
```

In [31]:
```
url = 'https://www.worldometers.info/geography/how-many-countries-are-there-in-the-world/'
#scraping the data of Countries in the World:
```

In [32]:
```
r = re.get(url) #Getting response form the website
print(r)        #If Respnse is 200, it is a valid response
```

```
<Response [200]>
```

In [33]:
```
 # parsing the response by using html.parser
world = bs(r.content, 'html.parser')
world
```

Out[33]:
```
<!DOCTYPE html>
<!--[if IE 8]> <html lang="en" class="ie8"> <![endif]--><!--[if IE 9]> <html lang="en" class="ie9"> <![endif]--><!--[if !IE]>
<!--> <html lang="en"> <!--<![endif]--> <head> <meta charset="utf-8"/> <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/> <title>How many countries are there in the world? (202
3) - Total &amp;amp; List | Worldometer</title><!-- Favicon --><link href="/favicon/favicon.ico" rel="shortcut icon" type="image/
x-icon"/><link href="/favicon/apple-icon-57x57.png" rel="apple-touch-icon" sizes="57x57"/><link href="/favicon/apple-icon-60x
60.png" rel="apple-touch-icon" sizes="60x60"/><link href="/favicon/apple-icon-72x72.png" rel="apple-touch-icon" sizes="72x7
2"/><link href="/favicon/apple-icon-76x76.png" rel="apple-touch-icon" sizes="76x76"/><link href="/favicon/apple-icon-114x114.
png" rel="apple-touch-icon" sizes="114x114"/><link href="/favicon/apple-icon-120x120.png" rel="apple-touch-icon" sizes="120x1
20"/><link href="/favicon/apple-icon-144x144.png" rel="apple-touch-icon" sizes="144x144"/><link href="/favicon/apple-icon-152
x152.png" rel="apple-touch-icon" sizes="152x152"/><link href="/favicon/apple-icon-180x180.png" rel="apple-touch-icon" sizes
="180x180"/><link href="/favicon/android-icon-192x192.png" rel="icon" sizes="192x192" type="image/png"/><link href="/favicon/
favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/><link href="/favicon/favicon-96x96.png" rel="icon" sizes="96x9
6" type="image/png"/><link href="/favicon/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/><link href="/favicon/
manifest.json" rel="manifest"/><meta content="#ffffff" name="msapplication-TileColor"/><meta content="/favicon/ms-icon-144x14
4.png" name="msapplication-TileImage"/><meta content="#ffffff" name="theme-color"/><!-- og image --><meta content="http://ww
w.worldometers.info/img/worldometers-fb.jpg" property="og:image"> <!-- Bootstrap --> <link href="/css/bootstrap.min.css" rel
="stylesheet"/><link href="/wm16.css" rel="stylesheet"/><!-- font awesome --><link href="https://maxcdn.bootstrapcdn.com/font
```

In [67]:
```
r1 = world.findAll('td')
print(len(r1))
```

```
975
```

In [68]:
```

d1 = world.select('td')[0].text
d2 = world.select('td')[1].text
d3 = world.select('td')[2].text
d4 = world.select('td')[3].text
d5 = world.select('td')[4].text
d6 = world.select('td')[5].text
d7 = world.select('td')[6].text
d8 = world.select('td')[7].text
# Printing the first 8 data entries from the 'td' tag
d1,d2,d3,d4,d5,d6,d7,d8
```

Out[68]:
```
('1',
 'China',
 '1,439,323,776',
 '18.5 %',
 '9,388,211',
 '2',
 'India',
 '1,380,004,385')
```

In [75]:
```
data=[] # Making an empty list
```

In [76]:
```
for i in range (0,len(r1),5):
        Sr_no = world.select('td')[i].text
        Country = world.select('td')[i+1].text
        Population_2020 = world.select('td')[i+2].text
        World_Share = world.select('td')[i+3].text
        Land_Area = world.select('td')[i+4].text
        data.append((
            Sr_no,Country,
            int(Population_2020.replace(',', '')),
            float(World_Share.replace('%', '')),
            int(Land_Area.replace(',', '')),))
```

In [77]:
```
print(data)
```

```
[('1', 'China', 1439323776, 18.5, 9388211), ('2', 'India', 1380004385, 17.7, 2973190), ('3', 'United States', 331002651, 4.2, 9
147420), ('4', 'Indonesia', 273523615, 3.5, 1811570), ('5', 'Pakistan', 220892340, 2.8, 770880), ('6', 'Brazil', 212559417, 2.
7, 8358140), ('7', 'Nigeria', 206139589, 2.6, 910770), ('8', 'Bangladesh', 164689383, 2.1, 130170), ('9', 'Russia', 145934462,
1.9, 16376870), ('10', 'Mexico', 128932753, 1.7, 1943950), ('11', 'Japan', 126476461, 1.6, 364555), ('12', 'Ethiopia', 11496358
8, 1.5, 1000000), ('13', 'Philippines', 109581078, 1.4, 298170), ('14', 'Egypt', 102334404, 1.3, 995450), ('15', 'Vietnam', 973
38579, 1.2, 310070), ('16', 'DR Congo', 89561403, 1.1, 2267050), ('17', 'Turkey', 84339067, 1.1, 769630), ('18', 'Iran', 839929
49, 1.1, 1628550), ('19', 'Germany', 83783942, 1.1, 348560), ('20', 'Thailand', 69799978, 0.9, 510890), ('21', 'United Kingdo
m', 67886011, 0.9, 241930), ('22', 'France', 65273511, 0.8, 547557), ('23', 'Italy', 60461826, 0.8, 294140), ('24', 'Tanzania',
59734218, 0.8, 885800), ('25', 'South Africa', 59308690, 0.8, 1213090), ('26', 'Myanmar', 54409800, 0.7, 653290), ('27', 'Keny
a', 53771296, 0.7, 569140), ('28', 'South Korea', 51269185, 0.7, 97230), ('29', 'Colombia', 50882891, 0.7, 1109500), ('30', 'Sp
ain', 46754778, 0.6, 498800), ('31', 'Uganda', 45741007, 0.6, 199810), ('32', 'Argentina', 45195774, 0.6, 2736690), ('33', 'Alg
eria', 43851044, 0.6, 2381740), ('34', 'Sudan', 43849260, 0.6, 1765048), ('35', 'Ukraine', 43733762, 0.6, 579320), ('36', 'Ira
q', 40222493, 0.5, 434320), ('37', 'Afghanistan', 38928346, 0.5, 652860), ('38', 'Poland', 37846611, 0.5, 306230), ('39', 'Cana
da', 37742154, 0.5, 9093510), ('40', 'Morocco', 36910560, 0.5, 446300), ('41', 'Saudi Arabia', 34813871, 0.4, 2149690), ('42',
'Uzbekistan', 33469203, 0.4, 425400), ('43', 'Peru', 32971854, 0.4, 1280000), ('44', 'Angola', 32866272, 0.4, 1246700), ('45',
'Malaysia', 32365999, 0.4, 328550), ('46', 'Mozambique', 31255435, 0.4, 786380), ('47', 'Ghana', 31072940, 0.4, 227540), ('48',
'Yemen', 29825964, 0.4, 527970), ('49', 'Nepal', 29136808, 0.4, 143350), ('50', 'Venezuela', 28435940, 0.4, 882050), ('51', 'Ma
dagascar', 27691018, 0.4, 581795), ('52', 'Cameroon', 26545863, 0.3, 472710), ('53', "Côte d'Ivoire", 26378274, 0.3, 318000),
('54', 'North Korea', 25778816, 0.3, 120410), ('55', 'Australia', 25499884, 0.3, 7682300), ('56', 'Niger', 24206644, 0.3, 12667
00), ('57', 'Sri Lanka', 21413249, 0.3, 62710), ('58', 'Burkina Faso', 20903273, 0.3, 273600), ('59', 'Mali', 20250833, 0.3, 12
20190), ('60', 'Romania', 19237691, 0.2, 230170), ('61', 'Malawi', 19129952, 0.2, 94280), ('62', 'Chile', 19116201, 0.2, 74353
2), ('63', 'Kazakhstan', 18776707, 0.2, 2699700), ('64', 'Zambia', 18383955, 0.2, 743390), ('65', 'Guatemala', 17915568, 0.2, 1
07160), ('66', 'Ecuador', 17643054, 0.2, 248360), ('67', 'Syria', 17500658, 0.2, 183630), ('68', 'Netherlands', 17134872, 0.2,
33720), ('69', 'Senegal', 16743927, 0.2, 192530), ('70', 'Cambodia', 16718965, 0.2, 176520), ('71', 'Chad', 16425864, 0.2, 1259
200), ('72', 'Somalia', 15893222, 0.2, 627340), ('73', 'Zimbabwe', 14862924, 0.2, 386850), ('74', 'Guinea', 13132795, 0.2, 2457
20), ('75', 'Rwanda', 12952218, 0.2, 24670), ('76', 'Benin', 12123200, 0.2, 112760), ('77', 'Burundi', 11890784, 0.2, 25680),
('78', 'Tunisia', 11818619, 0.2, 155360), ('79', 'Bolivia', 11673021, 0.1, 1083300), ('80', 'Belgium', 11589623, 0.1, 30280),
('81', 'Haiti', 11402528, 0.1, 27560), ('82', 'Cuba', 11326616, 0.1, 106440), ('83', 'South Sudan', 11193725, 0.1, 610952), ('8
4', 'Dominican Republic', 10847910, 0.1, 48320), ('85', 'Czech Republic (Czechia)', 10708981, 0.1, 77240), ('86', 'Greece', 104
23054, 0.1, 128900), ('87', 'Jordan', 10203134, 0.1, 88780), ('88', 'Portugal', 10196709, 0.1, 91590), ('89', 'Azerbaijan', 101
39177, 0.1, 82658), ('90', 'Sweden', 10099265, 0.1, 410340), ('91', 'Honduras', 9904607, 0.1, 111890), ('92', 'United Arab Emir
ates', 9890402, 0.1, 83600), ('93', 'Hungary', 9660351, 0.1, 90530), ('94', 'Tajikistan', 9537645, 0.1, 139960), ('95', 'Belaru
s', 9449323, 0.1, 202910), ('96', 'Austria', 9006398, 0.1, 82409), ('97', 'Papua New Guinea', 8947024, 0.1, 452860), ('98', 'Se
rbia', 8737371, 0.1, 87460), ('99', 'Israel', 8655535, 0.1, 21640), ('100', 'Switzerland', 8654622, 0.1, 39516), ('101', 'Tog
o', 8278724, 0.1, 54390), ('102', 'Sierra Leone', 7976983, 0.1, 72180), ('103', 'Laos', 7275560, 0.1, 230800), ('104', 'Paragua
y', 7132538, 0.1, 397300), ('105', 'Bulgaria', 6948445, 0.1, 108560), ('106', 'Libya', 6871292, 0.1, 1759540), ('107', 'Lebano
n', 6825445, 0.1, 10230), ('108', 'Nicaragua', 6624554, 0.1, 120340), ('109', 'Kyrgyzstan', 6524195, 0.1, 191800), ('110', 'El
Salvador', 6486205, 0.1, 20720), ('111', 'Turkmenistan', 6031200, 0.1, 469930), ('112', 'Singapore', 5850342, 0.1, 700), ('11
3', 'Denmark', 5792202, 0.1, 42430), ('114', 'Finland', 5540720, 0.1, 303890), ('115', 'Congo', 5518087, 0.1, 341500), ('116',
'Slovakia', 5459642, 0.1, 48088), ('117', 'Norway', 5421241, 0.1, 365268), ('118', 'Oman', 5106626, 0.1, 309500), ('119', 'Stat
e of Palestine', 5101414, 0.1, 6020), ('120', 'Costa Rica', 5094118, 0.1, 51060), ('121', 'Liberia', 5057681, 0.1, 96320), ('12
2', 'Ireland', 4937786, 0.1, 68890), ('123', 'Central African Republic', 4829767, 0.1, 622980), ('124', 'New Zealand', 4822233,
0.1, 263310), ('125', 'Mauritania', 4649658, 0.1, 1030700), ('126', 'Panama', 4314767, 0.1, 74340), ('127', 'Kuwait', 4270571,
0.1, 17820), ('128', 'Croatia', 4105267, 0.1, 55960), ('129', 'Moldova', 4033963, 0.1, 32850), ('130', 'Georgia', 3989167, 0.1,
69490), ('131', 'Eritrea', 3546421, 0.0, 101000), ('132', 'Uruguay', 3473730, 0.0, 175020), ('133', 'Bosnia and Herzegovina', 3
280819, 0.0, 51000), ('134', 'Mongolia', 3278290, 0.0, 1553560), ('135', 'Armenia', 2963243, 0.0, 28470), ('136', 'Jamaica', 29
61167, 0.0, 10830), ('137', 'Qatar', 2881053, 0.0, 11610), ('138', 'Albania', 2877797, 0.0, 27400), ('139', 'Lithuania', 272228
9, 0.0, 62674), ('140', 'Namibia', 2540905, 0.0, 823290), ('141', 'Gambia', 2416668, 0.0, 10120), ('142', 'Botswana', 2351627,
0.0, 566730), ('143', 'Gabon', 2225734, 0.0, 257670), ('144', 'Lesotho', 2142249, 0.0, 30360), ('145', 'North Macedonia', 20833
74, 0.0, 25220), ('146', 'Slovenia', 2078938, 0.0, 20140), ('147', 'Guinea-Bissau', 1968001, 0.0, 28120), ('148', 'Latvia', 188
6198, 0.0, 62200), ('149', 'Bahrain', 1701575, 0.0, 760), ('150', 'Equatorial Guinea', 1402985, 0.0, 28050), ('151', 'Trinidad
and Tobago', 1399488, 0.0, 5130), ('152', 'Estonia', 1326535, 0.0, 42390), ('153', 'Timor-Leste', 1318445, 0.0, 14870), ('154',
'Mauritius', 1271768, 0.0, 2030), ('155', 'Cyprus', 1207359, 0.0, 9240), ('156', 'Eswatini', 1160164, 0.0, 17200), ('157', 'Dji
bouti', 988000, 0.0, 23180), ('158', 'Fiji', 896445, 0.0, 18270), ('159', 'Comoros', 869601, 0.0, 1861), ('160', 'Guyana', 7865
52, 0.0, 196850), ('161', 'Bhutan', 771608, 0.0, 38117), ('162', 'Solomon Islands', 686884, 0.0, 27990), ('163', 'Montenegro',
628066, 0.0, 13450), ('164', 'Luxembourg', 625978, 0.0, 2590), ('165', 'Suriname', 586632, 0.0, 156000), ('166', 'Cabo Verde',
555987, 0.0, 4030), ('167', 'Micronesia', 548914, 0.0, 700), ('168', 'Maldives', 540544, 0.0, 300), ('169', 'Malta', 441543, 0.
0, 320), ('170', 'Brunei ', 437479, 0.0, 5270), ('171', 'Belize', 397628, 0.0, 22810), ('172', 'Bahamas', 393244, 0.0, 10010),
('173', 'Iceland', 341243, 0.0, 100250), ('174', 'Vanuatu', 307145, 0.0, 12190), ('175', 'Barbados', 287375, 0.0, 430), ('176',
'Sao Tome & Principe', 219159, 0.0, 960), ('177', 'Samoa', 198414, 0.0, 2830), ('178', 'Saint Lucia', 183627, 0.0, 610), ('17
9', 'Kiribati', 119449, 0.0, 810), ('180', 'Grenada', 112523, 0.0, 340), ('181', 'St. Vincent & Grenadines', 110940, 0.0, 390),
('182', 'Tonga', 105695, 0.0, 720), ('183', 'Seychelles', 98347, 0.0, 460), ('184', 'Antigua and Barbuda', 97929, 0.0, 440),
('185', 'Andorra', 77265, 0.0, 470), ('186', 'Dominica', 71986, 0.0, 750), ('187', 'Marshall Islands', 59190, 0.0, 180), ('18
8', 'Saint Kitts & Nevis', 53199, 0.0, 260), ('189', 'Monaco', 39242, 0.0, 1), ('190', 'Liechtenstein', 38128, 0.0, 160), ('19
1', 'San Marino', 33931, 0.0, 60), ('192', 'Palau', 18094, 0.0, 460), ('193', 'Tuvalu', 11792, 0.0, 30), ('194', 'Nauru', 1082
4, 0.0, 20), ('195', 'Holy See', 801, 0.0, 0)]
```

In [79]:
```
# create DataFrame using data by defining thecolumn headers
df = pd.DataFrame(data, columns =['Sr_no','Country','Population(2020)','World_Share(%)','Land_Area_Km²'])
```

In [80]:
```
df.head() #Showing top 5 entries from the data
```

Out[80]:

	Sr_no	Country	Population(2020)	World_Share(%)	Land_Area_Km²
0	1	China	1439323776	18.5	9388211
1	2	India	1380004385	17.7	2973190
2	3	United States	331002651	4.2	9147420
3	4	Indonesia	273523615	3.5	1811570
4	5	Pakistan	220892340	2.8	770880

In [81]:
```
# saving the dataframe
df.to_csv('WorldPopulation_2020_data.csv', index=False)
```

In [82]:
```
df.columns
```

Out[82]:
```
Index(['Sr_no', 'Country', 'Population(2020)', 'World_Share(%)',
       'Land_Area_Km²'],
      dtype='object')
```

In [106]:
```
df.shape #row  = 195 and columns = 5
```

Out[106]:
```
(195, 5)
```

In [102]:
```
df.isnull().sum() #there is no null values
```

Out[102]:
```
Sr_no              0
Country            0
Population(2020)   0
World_Share(%)     0
Land_Area_Km²      0
dtype: int64
```

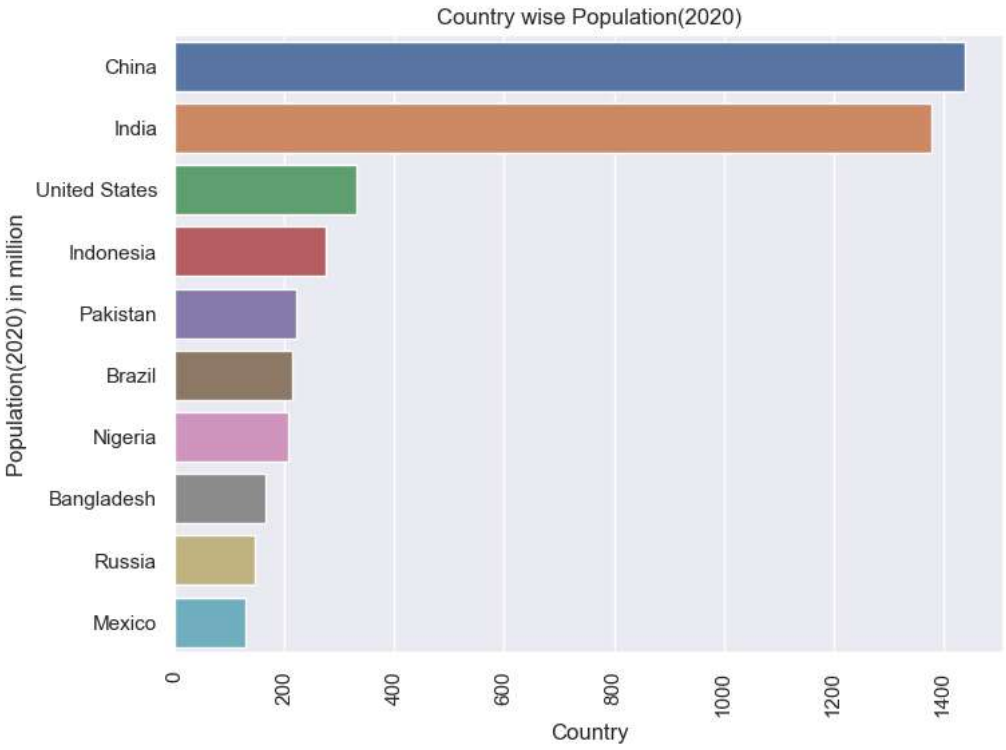
In [105]:
```
df.duplicated().sum()
```

Out[105]:
```
0
```

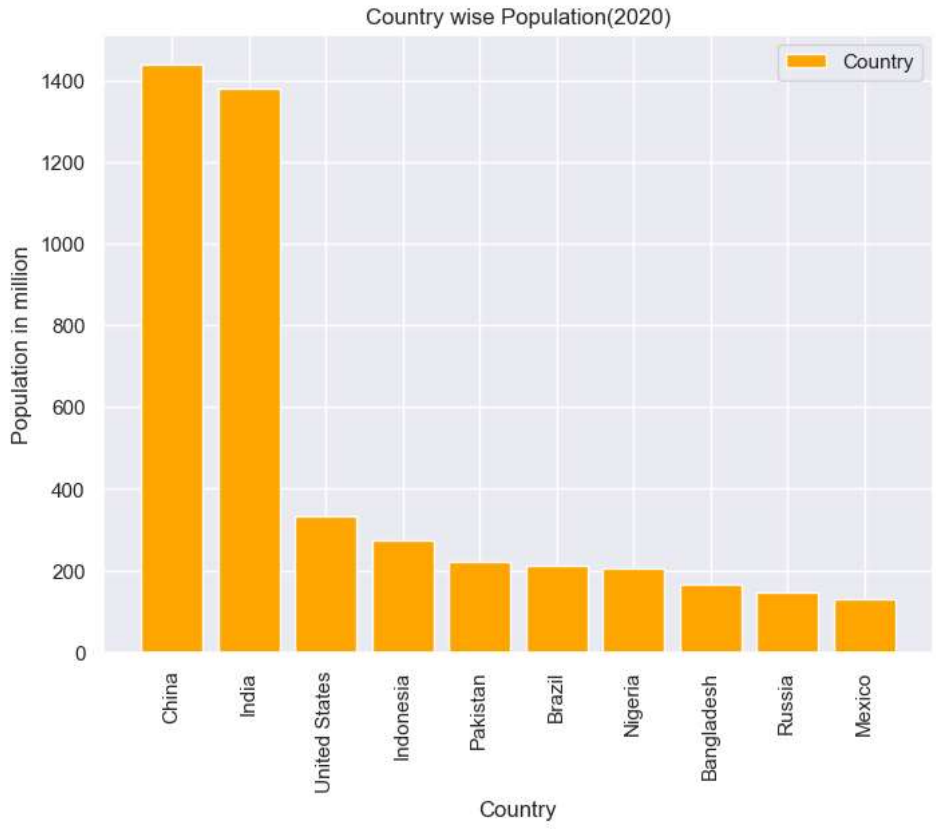
In [107]:
```
m = df['Population(2020)']/1000000
#as population value was not readable so that i have devided the population value by 1000000
```

In [94]:
```
## Analyzing the Data
```

In [116]:
```
### using seaborn
# Country vs Land_Area_Km² for top 10 countries by population
sns.set(rc={'figure.figsize':(8,6)})
sns.barplot(data = df, y = df['Country'].head(10), x = m.head(10))
plt.title('Country wise Population(2020)')
sns.set_style('darkgrid')
plt.xlabel('Country') # setting x-label
plt.ylabel('Population(2020) in million')  # setting y-label
plt.xticks(rotation=90)
# Showing the plot
plt.show()
```



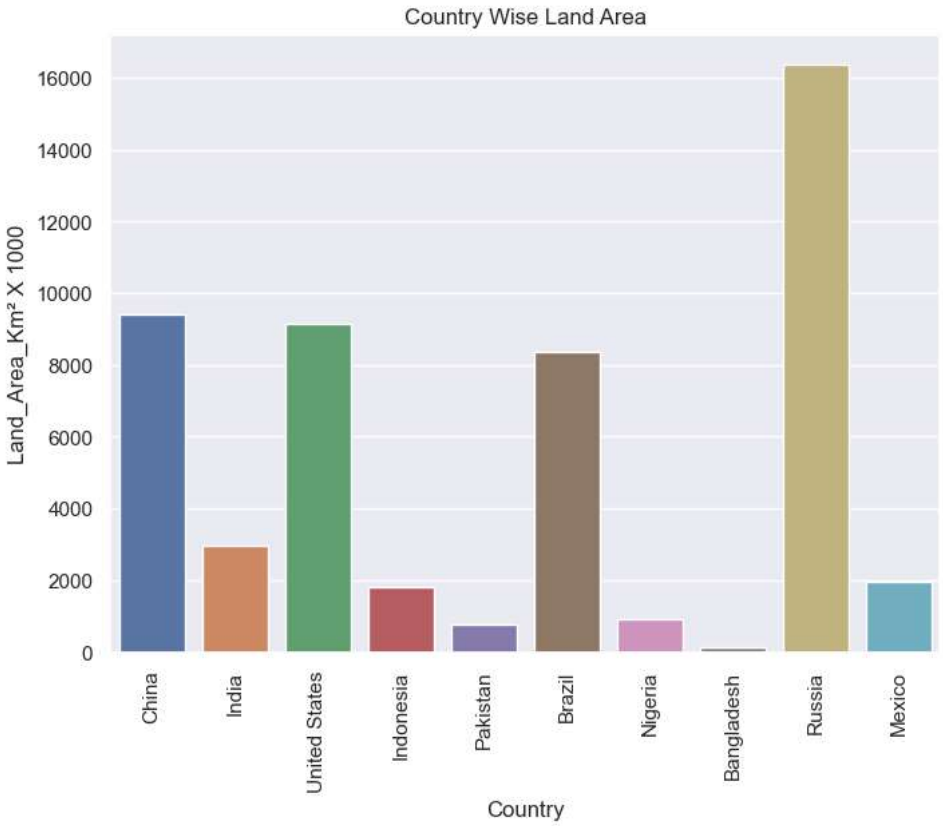
In [162]:
```
# Country vs Land_Area_Km² for top 10 countries by population
plt.figure(figsize=(8,6))
plt.bar(df['Country'].head(10), m.head(10), label = 'Country', color = 'orange/
')
plt.title('Country wise Population(2020)')
plt.xlabel('Country') # setting x-label
plt.ylabel('Population in million')  # setting y-label
plt.legend()
plt.xticks(rotation=90)
# Showing the plot
plt.show()
```



In [118]:
```
l = df['Land_Area_Km²']/1000
l.head()
```

Out[118]:
```
0    9388.211
1    2973.190
2    9147.420
3    1811.570
4     770.880
Name: Land_Area_Km², dtype: float64
```

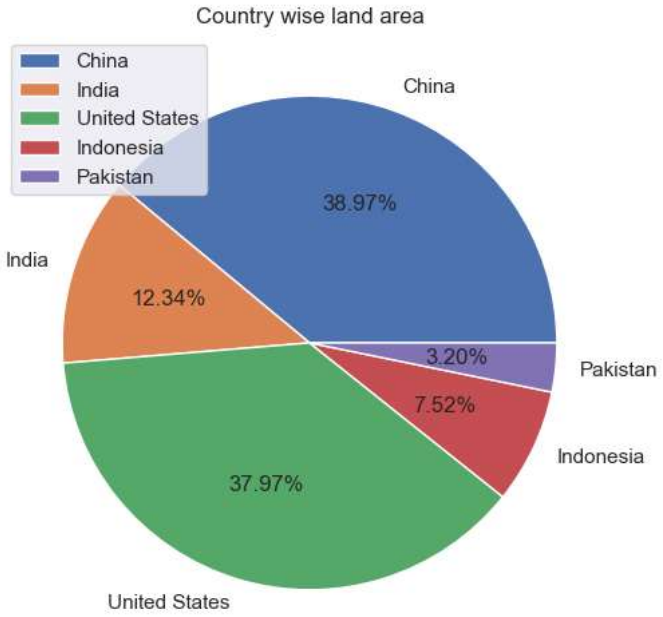
In [122]:
```
### using seaborn
# Population vs Land_Area_Km² for top 10 countries by population
sns.set(rc={'figure.figsize':(8,6)})
sns.barplot(data = df, x = df['Country'].head(10), y = l.head(10))
plt.title('Country Wise Land Area')
sns.set_style('darkgrid')
plt.xlabel('Country') # setting x-label
plt.ylabel('Land_Area_Km² X 1000')  # setting y-label
plt.xticks(rotation=90)
# Showing the plot
plt.show()
```



Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

In [161]:
```
plt.pie(df['Land_Area_Km²'].head(5), labels = df['Country'].head(5),autopct='%0.2f%%')
plt.title('Country wise land area')
plt.legend(loc = 2)
plt.show()
```



In [ ]:
```
```