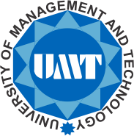
****

**NAME : MUHAMMAD HASHIM SHAFIQUE**

**ID : F2018065202**

**TO : MAZHAR JAVED AWAN**

**PROJECT DOCUMENTATION**

**Importing Necessary Libraries:**

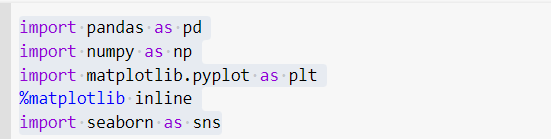
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

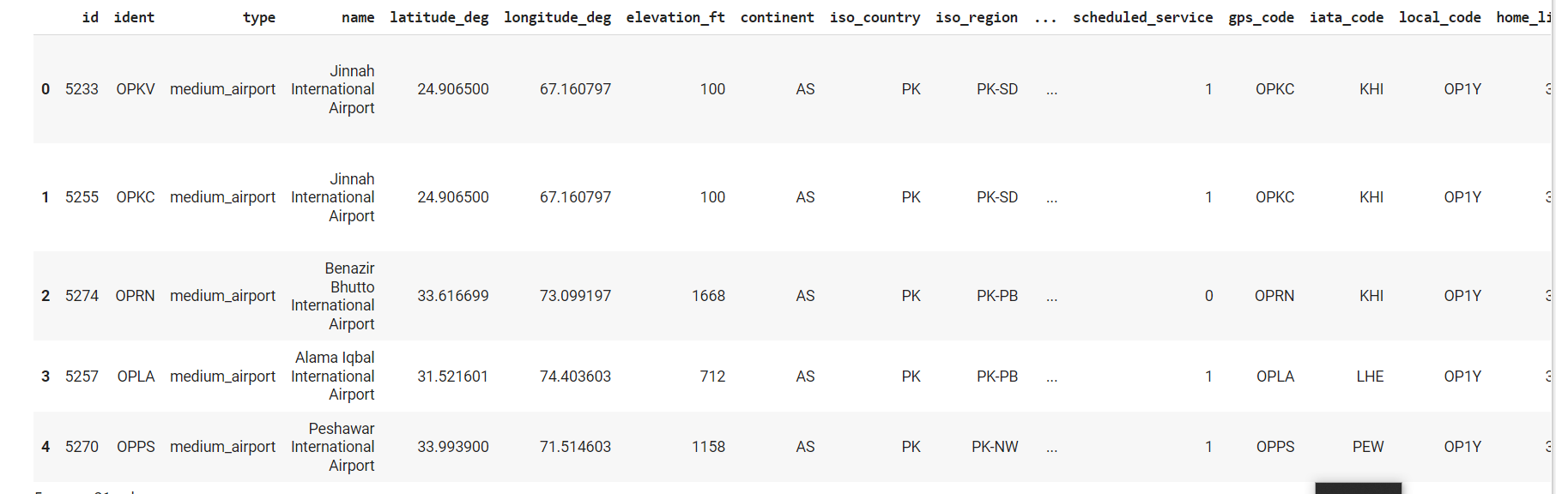
import seaborn as sns



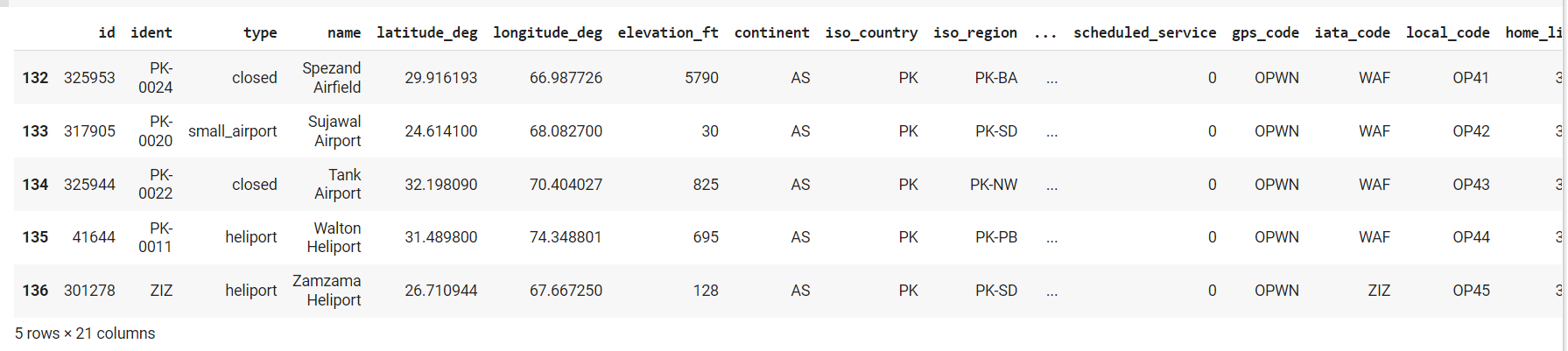
Previously my Dataset was UTF-8 encoded so had to convert into ISO-8859 Bytes of Encoding:

df = pd.read\_csv('/content/AirportsOfPakistan.csv', encoding=('ISO-8859-1'))

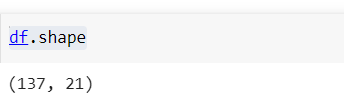
df.head()



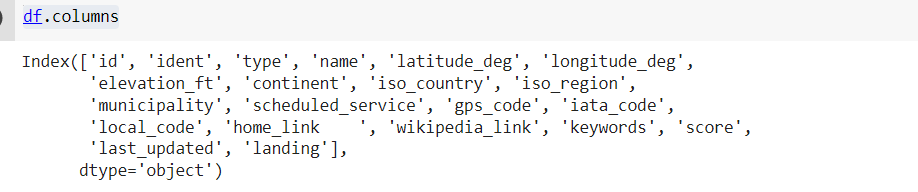
df.tail()



df.shape



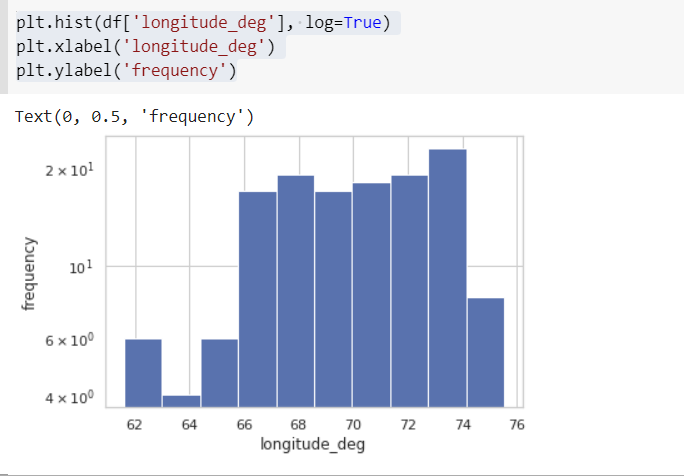
df.columns



plt.hist(df['longitude\_deg'], log=True)

plt.xlabel('longitude\_deg')

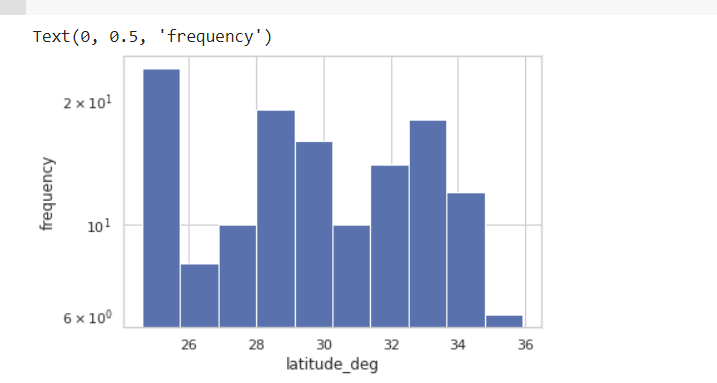
plt.ylabel('frequency')



plt.hist(df['latitude\_deg'], log=True)

plt.xlabel('latitude\_deg')

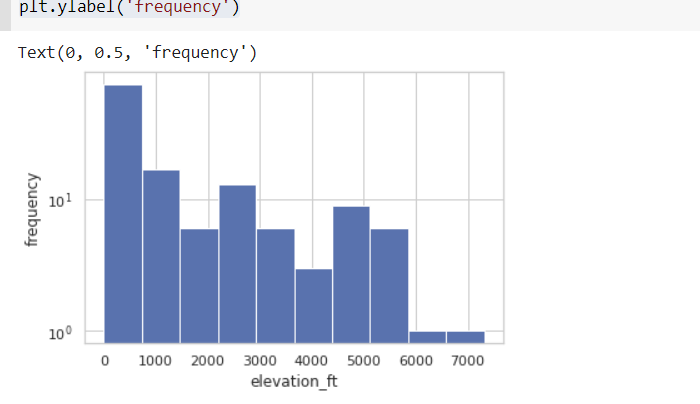
plt.ylabel('frequency')

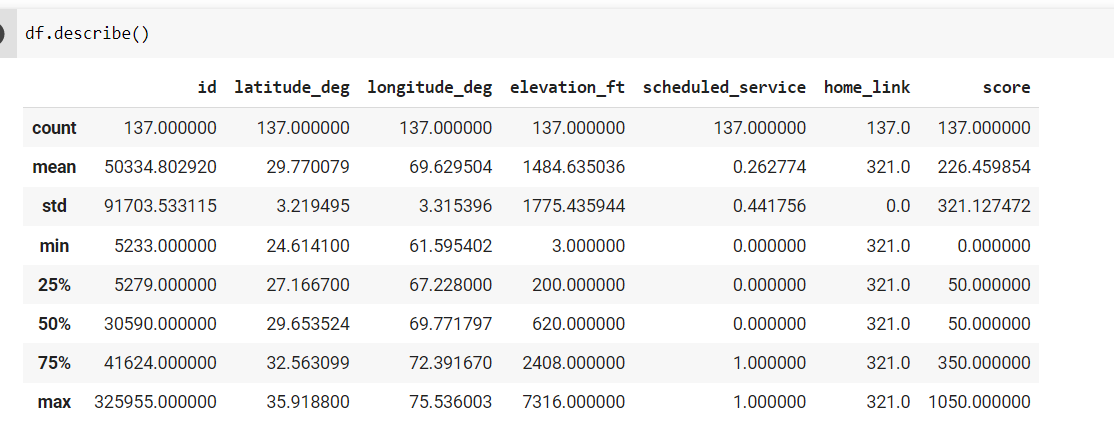


plt.hist(df['elevation\_ft'], log=True)

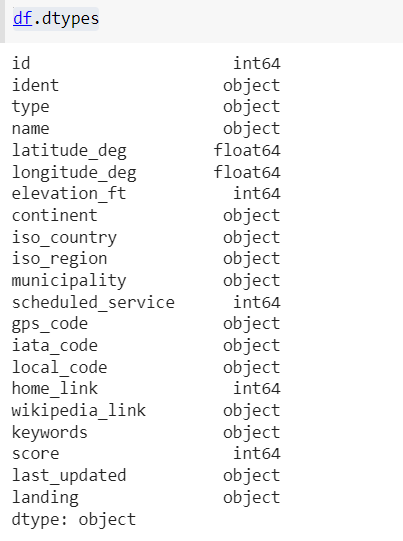
plt.xlabel('elevation\_ft')

plt.ylabel('frequency')

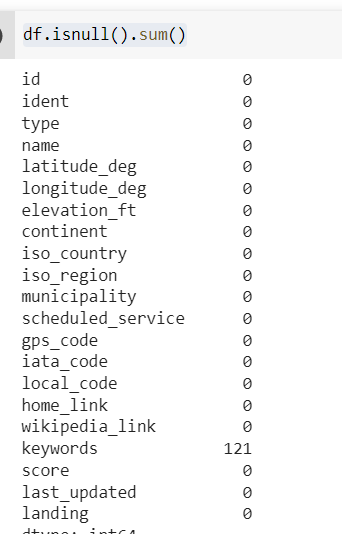


df.describe()

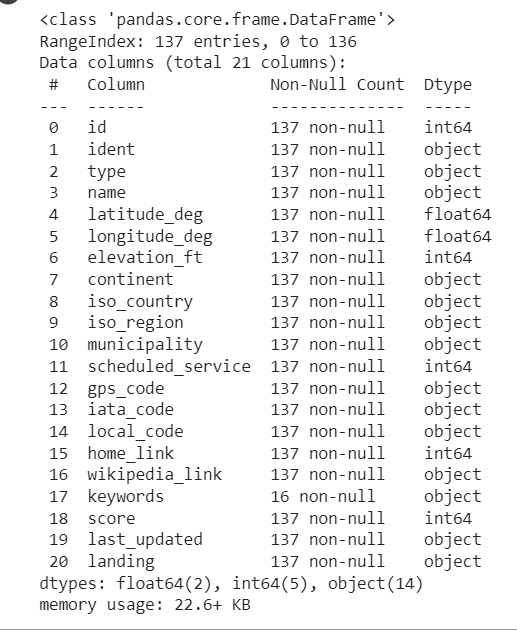
df.dtypes



df.isnull().sum()



df.info()

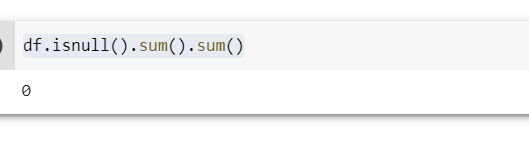


df = df.drop(['keywords'],axis=1)

df.isnull().sum()



df.isnull().sum().sum()

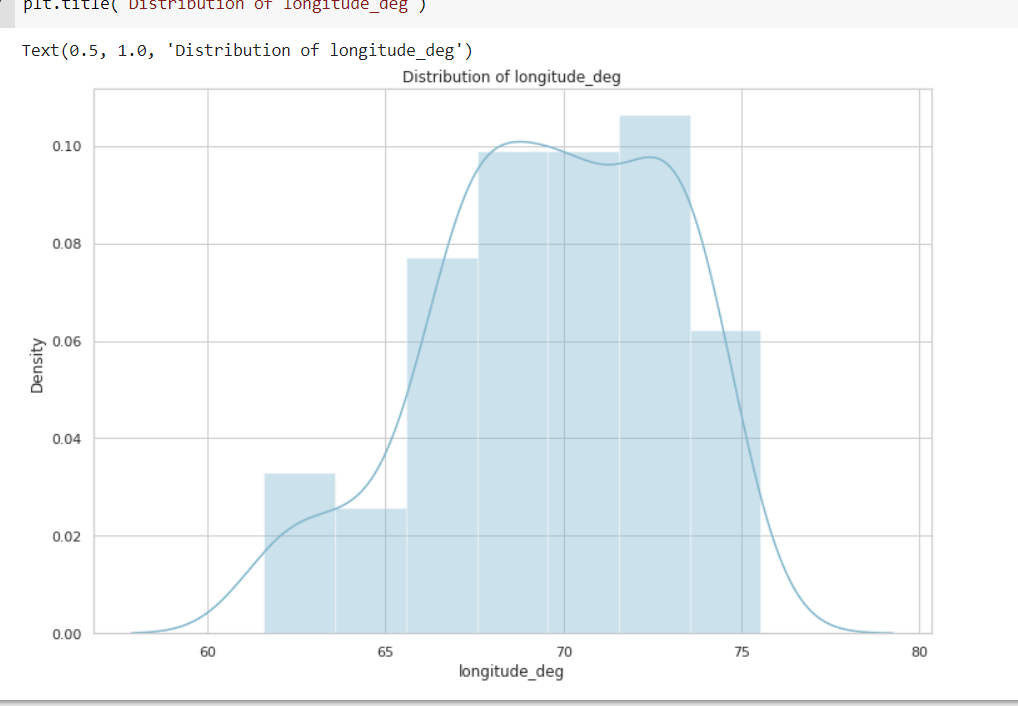


sns.set(style='whitegrid')

f, ax = plt.subplots(1,1, figsize=(12, 8))

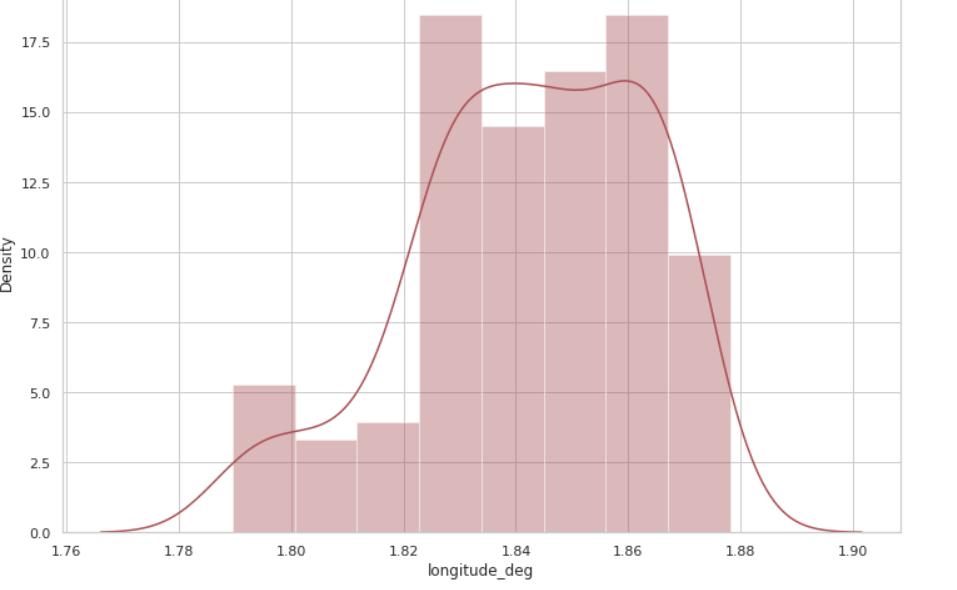
ax = sns.distplot(df['longitude\_deg'], kde = True, color = 'c')

plt.title('Distribution of longitude\_deg')

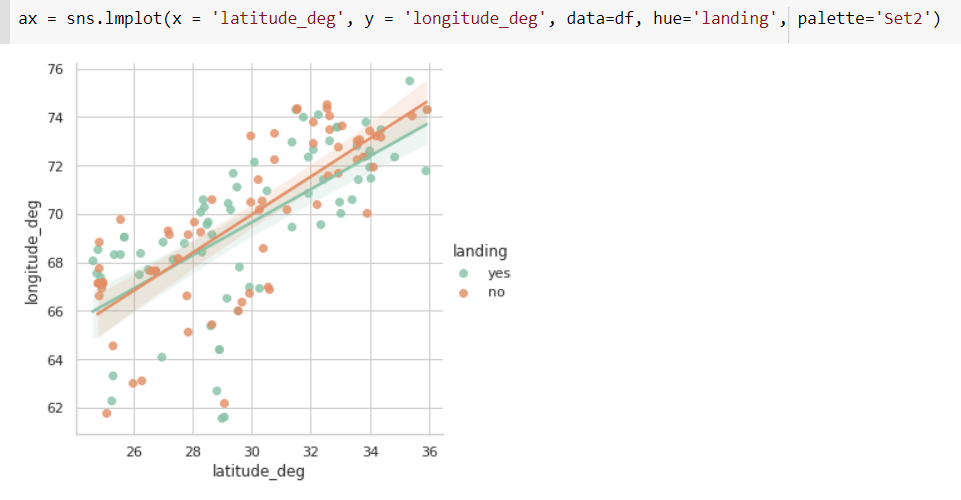


f, ax = plt.subplots(1, 1, figsize=(12, 8))

ax = sns.distplot(np.log10(df['longitude\_deg']), kde = True, color = 'r' )



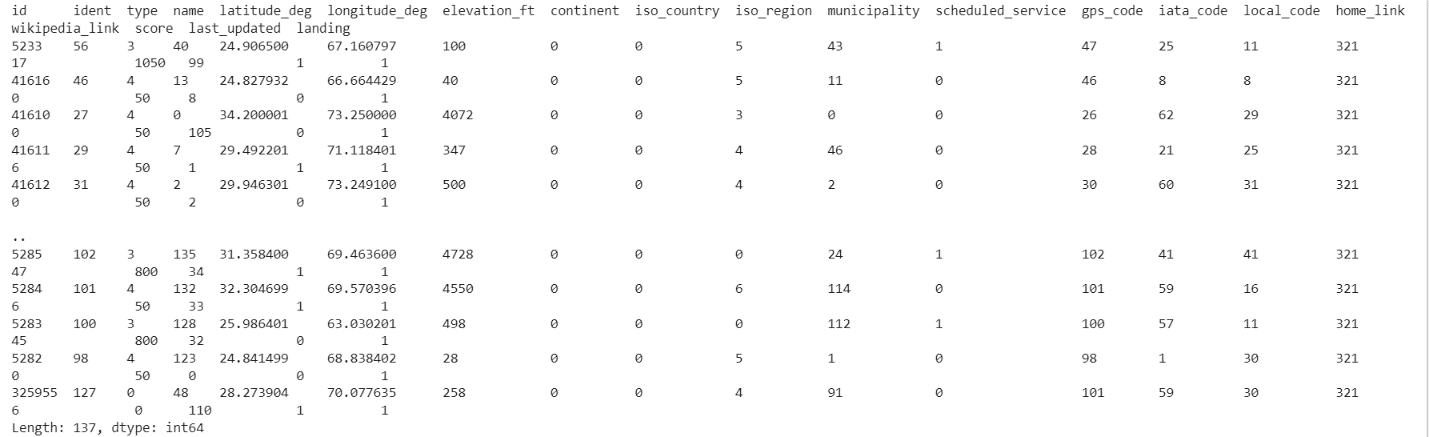
ax = sns.lmplot(x = 'latitude\_deg', y = 'longitude\_deg', data=df, hue='landing', palette='Set2')



df.memory\_usage()

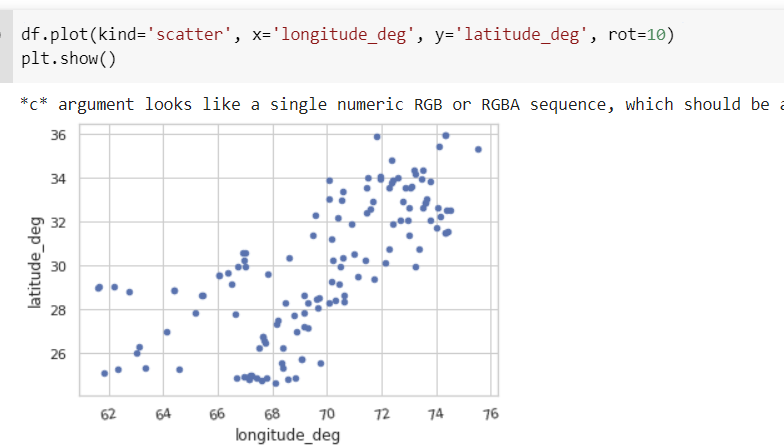


df.value\_counts()



df.plot(kind='scatter', x='longitude\_deg', y='latitude\_deg', rot=10)

plt.show()



x = df['longitude\_deg']

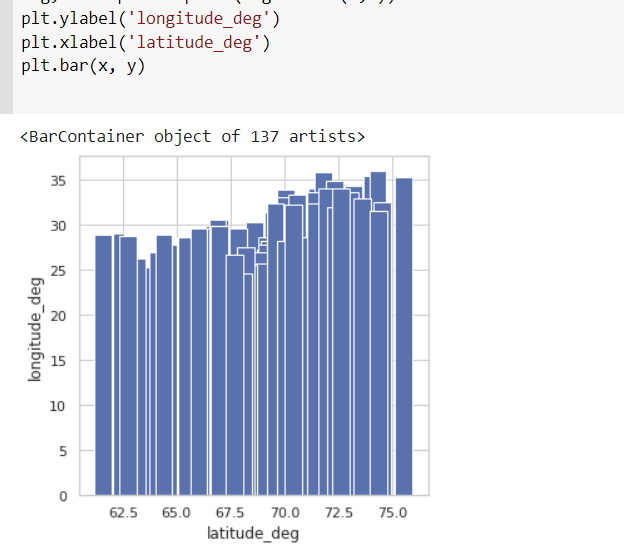
y = df['latitude\_deg']

fig, ax = plt.subplots(figsize = (5,5))

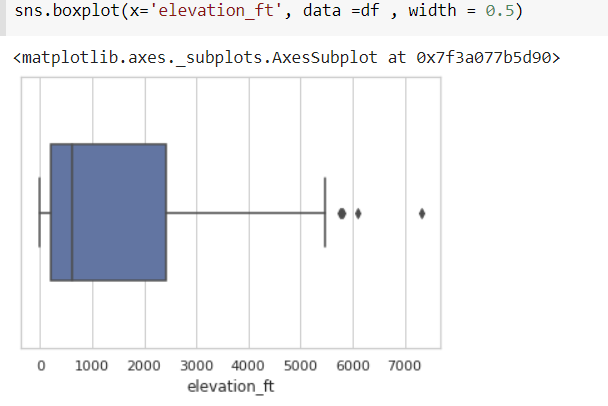
plt.ylabel('longitude\_deg')

plt.xlabel('latitude\_deg')

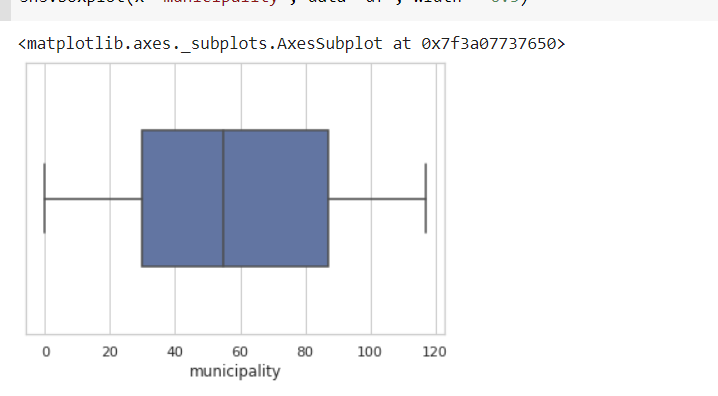
plt.bar(x, y)



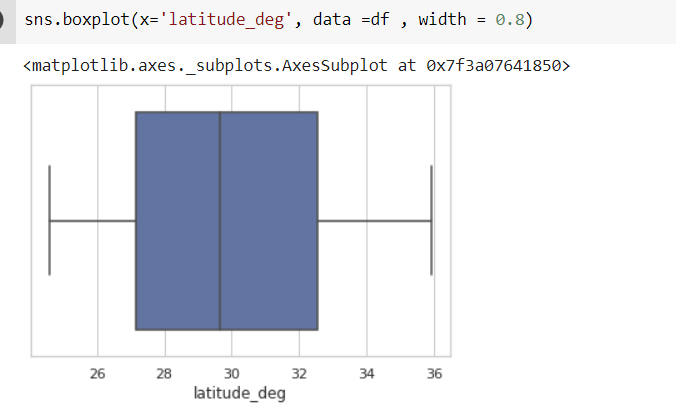
sns.boxplot(x='elevation\_ft', data =df , width = 0.5)



sns.boxplot(x='municipality', data =df , width = 0.5)



sns.boxplot(x='latitude\_deg', data =df , width = 0.8)



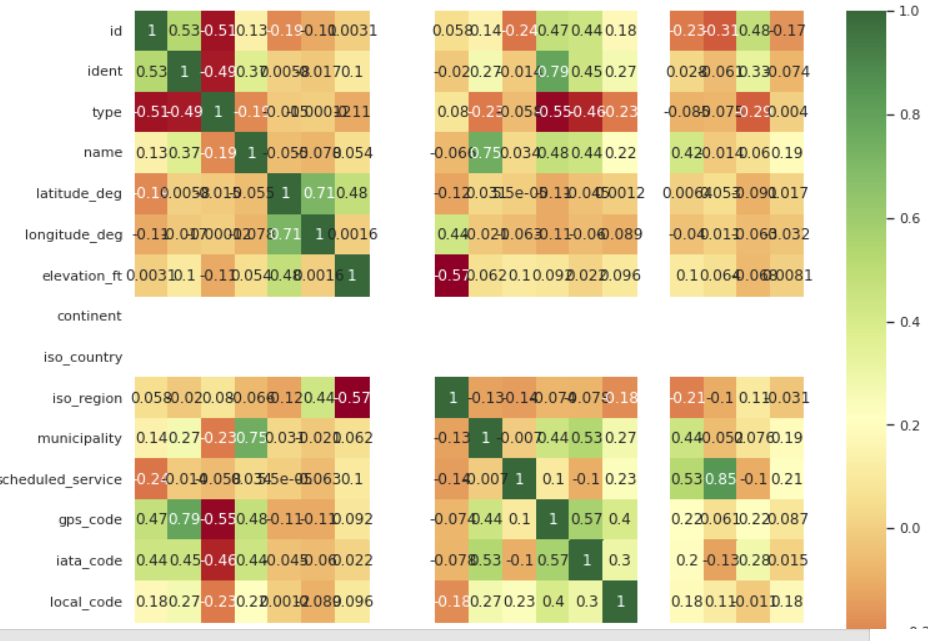
**FINDING CO-RELATIONS USING SEABORN HEATMAP:**

corrmat = df.corr()

top\_corr\_features = corrmat.index

plt.figure(figsize=(12,12))

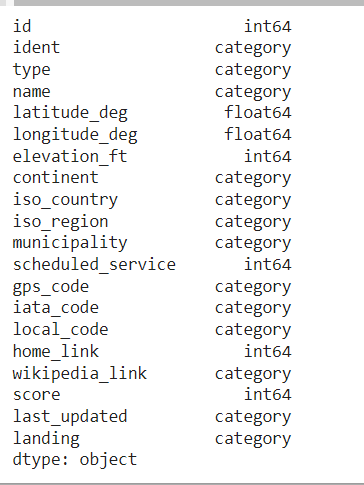
g=sns.heatmap(df.corr(),annot=True,cmap="blygcn")



**Linear Regression:**

df[['ident', 'type', 'name', 'continent', 'iso\_country', 'iso\_region', 'municipality', 'gps\_code', 'iata\_code', 'local\_code', 'wikipedia\_link', 'last\_updated','landing']] = df[['ident', 'type', 'name', 'continent', 'iso\_country', 'iso\_region', 'municipality', 'gps\_code', 'iata\_code', 'local\_code', 'wikipedia\_link', 'last\_updated','landing']].astype('category')

df.dtypes



##Converting category labels into numerical using LabelEncoder

from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()

label.fit(df.ident.drop\_duplicates())

df.ident = label.transform(df.ident)

label.fit(df.type.drop\_duplicates())

df.type = label.transform(df.type)

label.fit(df.name.drop\_duplicates())

df.name = label.transform(df.name)

label.fit(df.continent.drop\_duplicates())

df.continent = label.transform(df.continent)

label.fit(df.iso\_country.drop\_duplicates())

df.iso\_country = label.transform(df.iso\_country)

label.fit(df.iso\_region.drop\_duplicates())

df.iso\_region = label.transform(df.iso\_region)

label.fit(df.municipality.drop\_duplicates())

df.municipality = label.transform(df.municipality)

label.fit(df.gps\_code.drop\_duplicates())

df.gps\_code = label.transform(df.gps\_code)

label.fit(df.iata\_code.drop\_duplicates())

df.iata\_code = label.transform(df.iata\_code)

label.fit(df.local\_code.drop\_duplicates())

df.local\_code = label.transform(df.local\_code)

label.fit(df.wikipedia\_link.drop\_duplicates())

df.wikipedia\_link = label.transform(df.wikipedia\_link)

label.fit(df.last\_updated.drop\_duplicates())

df.last\_updated = label.transform(df.last\_updated)

label.fit(df.landing.drop\_duplicates())

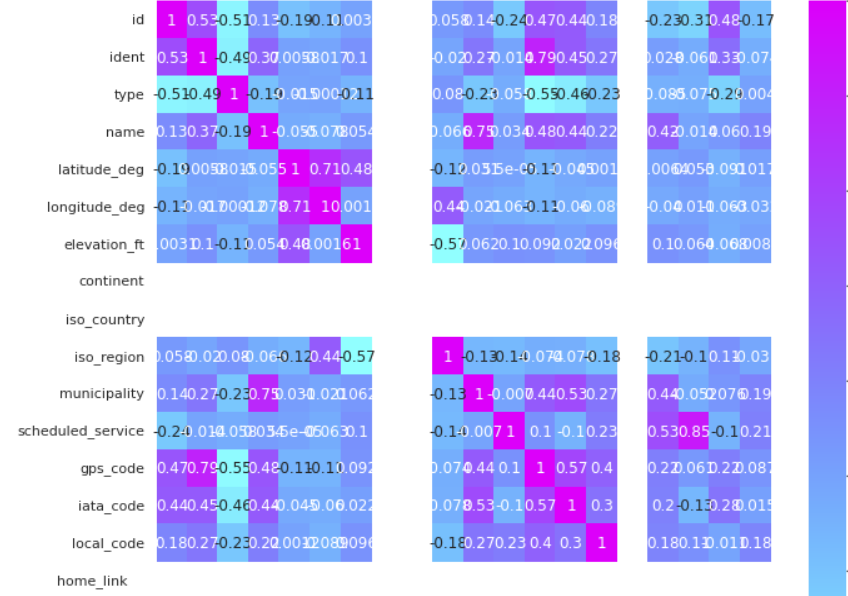
df.landing = label.transform(df.landing)

df.dtypes



f, ax = plt.subplots(1, 1, figsize=(11, 11))

ax = sns.heatmap(df.corr(), annot=True, cmap='cool')



from sklearn.linear\_model import LinearRegression

from sklearn import metrics

x = df.drop(['longitude\_deg'], axis = 1)

y = df['longitude\_deg']

x\_train, x\_test, y\_train, y\_test = holdout(x, y, test\_size=0.2, random\_state=0)

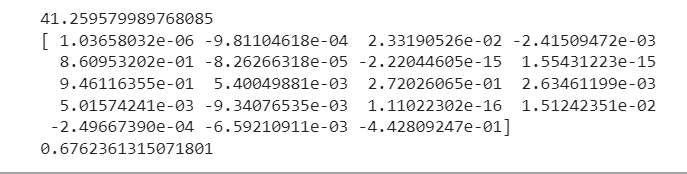
Linearreg = LinearRegression()

Linearreg.fit(x\_train, y\_train)

print(Lin\_reg.intercept\_)

print(Linearreg.coef\_)

print(Linearreg.score(x\_test, y\_test))



RANDOM FOREST REGRESSOR:

from sklearn.ensemble import RandomForestRegressor as r

x = df.drop(['longitude\_deg'], axis=1)

y = df.longitude\_deg

rf = r(n\_estimators = 100, criterion = 'mse',

                              random\_state = 1,

                              n\_jobs = -1)

rf.fit(x\_train,y\_train)

x\_train\_pred = Rfr.predict(x\_train)

x\_test\_pred = Rfr.predict(x\_test)

print('train data: %.3f, MSE test data: %.3f' %

      (metrics.mean\_squared\_error(x\_train\_pred, y\_train)

      ,metrics.mean\_squared\_error(x\_test\_pred, y\_test)))

print('2nd train data: %.3f,  test data: %.3f' %

      (metrics.r2score(y\_train,x\_train\_pred),

       metrics.r2score(y\_test,x\_test\_pred)))



print(classification\_report(y\_test, y\_predict, target\_names=class\_names)) print(confusion\_matrix(y\_test, y\_predict))

**🡨------------------------------------------------🡪**