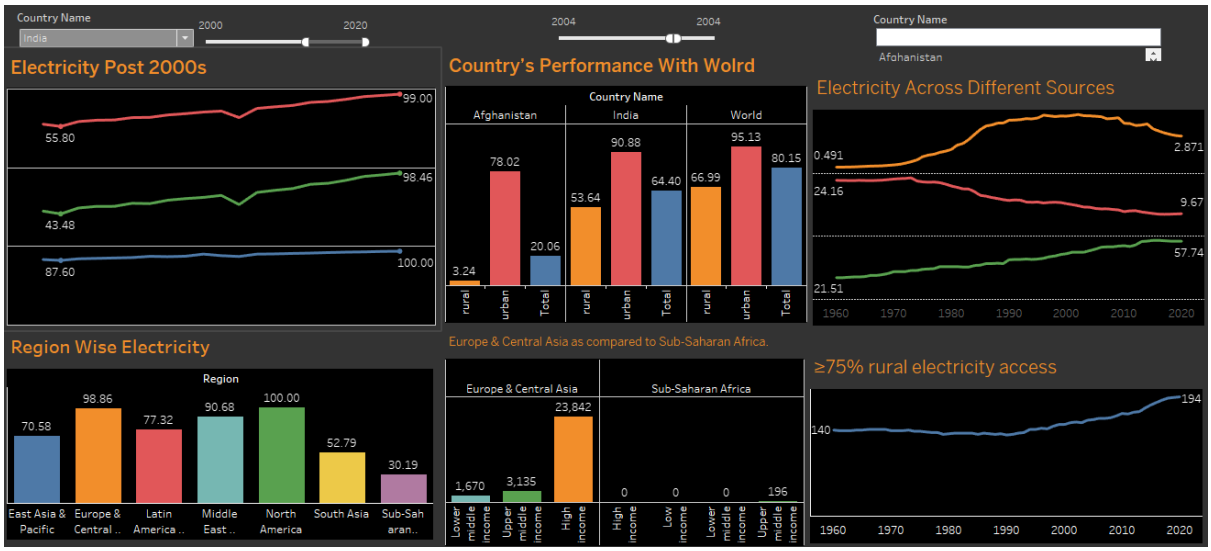
Tableau

https://public.tableau.com/views/WorldElecricityAnalysis/Dashboard1?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link



SQL Query

```
select A.Access_to_electricity_of_population,A.Years,B.CountryName from
[dbo].[dataLongForm]as A left join [dbo].[Country]as B on
A.Country_Code=B.Country_Code  where Years=2000
```

```
------Q-2--------------
select B.CountryName,B.Region,avg(A.Access_to_electricity_of_population)as avg_electricity from
[dbo].[dataLongForm]as A left join [dbo].[Country]as B on
A.Country_Code=B.Country_Code group by B.CountryName,B.Region
```

```
 ------Q-3--------------
 select B.CountryName,avg(A.Access_to_electricity_of_population)as avg_electricity,
 (select avg(Access_to_electricity_of_population)as avg_electricity from  [dbo].[dataLongForm]
 where Country_Code='wld')as world_avg
 from
 [dbo].[dataLongForm]as A left join [dbo].[Country]as B on
 A.Country_Code=B.Country_Code group by B.CountryName
```

```
------------Q-4----------------------
select distinct A.Years,
count(A.Access_to_electricity_rural_of_rural_population)over(partition by A.years) from
[dbo].[dataLongForm]as A left join [dbo].[Country]as B on
A.Country_Code=B.Country_Code where A.Access_to_electricity_rural_of_rural_population>=75 order by Years
```

```
----------Q-5--------------
select region,IncomeGroup,avg(B.[Electricity_production_from_nuclear_sources_of_total])as Nuclear
from [dbo].[Country]as A left join [dbo].[dataLongForm]as B on A.Country_Code=B.Country_Code
Where A.Region='Europe & Central Asia' or A.Region='Sub-Saharan Africa' group by region,IncomeGroup
```

```
---------Q-6

select [Years],avg(Electricity_production_from_nuclear_sources_of_total)as Nuclear
,avg(Electricity_production_from_oil_sources_of_total)as oil,
avg(Electricity_production_from_renewable_sources_excluding_hydroelectric_kWh)as renewable
 from [dbo].[dataLongForm] group by[Years] order by [Years]
```

Python (Pandas)

```
import pandas as pd
import numpy as np
from scipy.interpolate import CubicSpline
from scipy.interpolate import interp1d
import math
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn import linear_model
import sys
np.set_printoptions(threshold=sys.maxsize)
```

```
mainFolderLoc = r'E:\data science\New folder\world_bank_electricity_report'
dataFolderNames = ['access_to_elec_rural__world','access_to_electricity__total',
                   'access_to_electricity_urban__world',
                   'electricity_power_transmissoin_distribution_losses',
                   'electricity_production_nuclear_sources',
                   'electricity_production_oil_sources',
                   'electricity_production_renewable_sources']
jsonFileNames = ['API_EG.ELC.ACCS.RU.ZS_DS2_en_csv_v2_4364070',
                 'API_EG.ELC.ACCS.ZS_DS2_en_csv_v2_4353549',
                 'API_EG.ELC.ACCS.UR.ZS_DS2_en_csv_v2_4367536',
                 'API_EG.ELC.LOSS.ZS_DS2_en_csv_v2_4357259',
                 'API_EG.ELC.NUCL.ZS_DS2_en_csv_v2_4357319',
                 'API_EG.ELC.PETR.ZS_DS2_en_csv_v2_4356435',
                 'API_EG.ELC.RNWX.KH_DS2_en_csv_v2_4356007']
CountryMetaDataFiles = ['Metadata_Country_API_EG.ELC.ACCS.RU.ZS_DS2_en_csv_v2_4364070',
                        'Metadata_Country_API_EG.ELC.ACCS.ZS_DS2_en_csv_v2_4353549',
                        'Metadata_Country_API_EG.ELC.ACCS.UR.ZS_DS2_en_csv_v2_4367536',
                        'Metadata_Country_API_EG.ELC.LOSS.ZS_DS2_en_csv_v2_4357259',
                        'Metadata_Country_API_EG.ELC.NUCL.ZS_DS2_en_csv_v2_4357319',
                        'Metadata_Country_API_EG.ELC.PETR.ZS_DS2_en_csv_v2_4356435',
                        'Metadata_Country_API_EG.ELC.RNWX.KH_DS2_en_csv_v2_4356007']
IndicatorMetaDataFiles = ['Metadata_Indicator_API_EG.ELC.ACCS.RU.ZS_DS2_en_csv_v2_4364070',
                        'Metadata_Indicator_API_EG.ELC.ACCS.ZS_DS2_en_csv_v2_4353549',
                        'Metadata_Indicator_API_EG.ELC.ACCS.UR.ZS_DS2_en_csv_v2_4367536',
                        'Metadata_Indicator_API_EG.ELC.LOSS.ZS_DS2_en_csv_v2_4357259',
                        'Metadata_Indicator_API_EG.ELC.NUCL.ZS_DS2_en_csv_v2_4357319',
                        'Metadata_Indicator_API_EG.ELC.PETR.ZS_DS2_en_csv_v2_4356435',
                        'Metadata_Indicator_API_EG.ELC.RNWX.KH_DS2_en_csv_v2_4356007']
```

Cleaning data and filling missing values

```
def json_to_csv(json_loc,json_filename):
    df = pd.read_json(json_loc+'\\'+json_filename+'.json')
    data = preprocessData(df)
    data.to_csv(json_loc+'\\'+'data.csv', index = False)
    last_update = df.iloc[0][1]
    return data, last_update
```

```
def fillZero(data):
    data = data.reset_index(drop = True)
    for index, row in data.iterrows():
        if row.isnull().all():
            data.loc[index] = 0
    return data
```

```
def preprocessData(df):
    df.drop(columns=df.columns[-2:], axis=1, inplace=True)
    headers = df.iloc[1]
    headers = pd.to_numeric(headers, errors='ignore')
    data = df.iloc[2:].reset_index(drop=True)
    data.columns = headers
    #data = data.replace(np.nan,0)
    data1 = data[data.columns[4:]]
    data1 = data1.replace('',np.nan)
    data1 = fillZero(data1)
    x = np.array(data1.columns[:])
    x = x.astype(int)
    y = np.array(data1)
    y = y.astype(float)
    for j in range(len(y)):
        x1 = []
        y1 = []
        for i in range(len(y[j])):
            if not math.isnan(y[j][i]):
                x1.append(x[i])
                y1.append(y[j][i])
        linear = interp1d(x1,y1,fill_value="extrapolate")
        linearLimit = lambda x: 0 if x<0 else (100 if x>100 else x)
        y[j] = [linearLimit(k) for k in linear(x)]
    ydf =  pd.DataFrame(y,columns=x)
    mdf = data[data.columns[:4]]
    finalData = pd.concat([mdf,ydf],axis=1)
    #Remove not classified row
    finalData.drop(finalData.index[finalData['Country Code']=='INX'], axis=0 ,inplace = True)
    return finalData
```

```
rural_access, rural_access_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[0],jsonFileNames[0])
```

```
total_access, total_access_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[1],jsonFileNames[1])
```

```
urban_access, urban_access_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[2],jsonFileNames[2])
```

```
transmission_loss, transmission_loss_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[3],jsonFileNames[3])
```

```
nuclear_sources, nuclear_sources_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[4],jsonFileNames[4])
```

```
oil_sources, oil_sources_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[5],jsonFileNames[5])
```

```
renewable_sources, renewable_sources_lastcheck = json_to_csv(mainFolderLoc+'\\'+dataFolderNames[6],jsonFileNames[6])
```

Creating meta data tables

```
loc = mainFolderLoc+'\\access_to_elec_rural__world'
meta_country = pd.read_csv(loc+'\\'+'Metadata_Country_API_EG.ELC.ACCS.RU.ZS_DS2_en_csv_v2_4364070'+'.csv')
meta_indicator = pd.read_csv(loc+'\\'+'Metadata_Indicator_API_EG.ELC.ACCS.RU.ZS_DS2_en_csv_v2_4364070'+'.csv')

region = pd.DataFrame(meta_country['Region'].unique(),columns=['Region'])

Indicators = meta_indicator[meta_indicator.columns[:-1]]
Indicators['Last Update'] = rural_access_lastcheck

Country = meta_country[meta_country.columns[:-1]]
Country = Country.rename(columns={'TableName':'CountryName'})

Years = pd.DataFrame(np.arange(1960,2021),columns=['Years'])

dataLongForm = pd.melt(rural_access,id_vars=['Country Code'], value_vars=rural_access.columns[4:],var_name='Years',
        value_name=rural_access.iloc[0]['Indicator Name']).sort_values(['Country Code','Years'],ignore_index=True)

toLoc = mainFolderLoc
#
```

```
#regionOld = pd.read_csv(toLoc+'\\Region.csv')
#indicatorOld = pd.read_csv(toLoc+'\\Indicators.csv')
#countryOld = pd.read_csv(toLoc+'\\Country.csv')
#
#pd.concat([region,regionOld],ignore_index=True).drop_duplicates().to_csv(toLoc+'\\Region.csv', index=False)
#pd.concat([Indicators,indicatorOld],ignore_index=True).drop_duplicates().to_csv(toLoc+'\\Indicators.csv',index=False)
#pd.concat([Country,countryOld],ignore_index=True).drop_duplicates()

region.to_csv(toLoc+'\\Region.csv', index=False)
Indicators.to_csv(toLoc+'\\Indicators.csv',index=False)
Country.to_csv(toLoc+'\\Country.csv',index=False)
Years.to_csv(toLoc+'\\Years.csv',index=False)
dataLongForm.to_csv(toLoc+'\\dataLongForm.csv',index=False)
```

```
def convergeData(loc,filenameMetaCountry,filenameMetaIndicator):
    meta_country = pd.read_csv(loc+'\\'+filenameMetaCountry+'.csv')
    meta_indicator = pd.read_csv(loc+'\\'+filenameMetaIndicator+'.csv')

    toLocRegion = mainFolderLoc+'\\Region.csv'
    toLocIndicator = mainFolderLoc+'\\Indicators.csv'
    toLocCountry = mainFolderLoc+'\\Country.csv'
    toLocYears =mainFolderLoc+'\\Years.csv'
    toLocData = mainFolderLoc+'\\dataLongForm.csv'

    Region = pd.DataFrame(meta_country['Region'].unique(),columns=['Region'])

    Indicators = meta_indicator[meta_indicator.columns[:-1]]
    Indicators['Last Update'] = rural_access_lastcheck

    Country = meta_country[meta_country.columns[:-1]]
    Country = Country.rename(columns={'TableName':'CountryName'})

    regionOld = pd.read_csv(toLocRegion)
    indicatorOld = pd.read_csv(toLocIndicator)
    countryOld = pd.read_csv(toLocCountry)

    pd.concat([Region,regionOld],ignore_index=True).drop_duplicates().to_csv(toLocRegion, index=False)
    pd.concat([Indicators,indicatorOld],ignore_index=True).drop_duplicates().to_csv(toLocIndicator,index=False)
    pd.concat([Country,countryOld],ignore_index=True).drop_duplicates().to_csv(toLocCountry, index=False)
```

```
for i in range(7):
    convergeData(mainFolderLoc+'\\'+dataFolderNames[i],CountryMetaDataFiles[i],IndicatorMetaDataFiles[i])
```

```
def dataLongForm(mainFolderLoc, dataFolderNames):
    for i in range(7):
        data = pd.read_csv(mainFolderLoc+'\\'+dataFolderNames[i]+'\\data.csv')
        dataOld = pd.read_csv(mainFolderLoc+'\\'+'dataLongForm.csv')

        dataLongForm = pd.melt(data,id_vars=['Country Code'], value_vars=data.columns[4:],var_name='Years',
        value_name=data.iloc[0]['Indicator Name']).sort_values(['Country Code','Years'],ignore_index=True)

        dataOld[data.iloc[0]['Indicator Name']] = dataLongForm[data.iloc[0]['Indicator Name']]
        dataOld.to_csv(mainFolderLoc+'\\'+'dataLongForm.csv', index=False)
    return dataOld
```

```
dataLongForm(mainFolderLoc, dataFolderNames)
```

	Country Code	Years	Access to electricity, rural (% of rural population)	Access to electricity (% of population)	Access to electricity, urban (% of urban population)	Electric power transmission and distribution losses (% of output)	Electricity production from nuclear sources (% of total)	Electricity production from oil sources (% of total)	Electricity production from renewable sources, excluding hydroelectric (kWh)
0	ABW	1960	100.000000	100.000000	100.000000	0.000000	0.0	0.000000	0.0
1	ABW	1961	100.000000	100.000000	100.000000	0.000000	0.0	0.000000	0.0
2	ABW	1962	100.000000	100.000000	100.000000	0.000000	0.0	0.000000	0.0
3	ABW	1963	100.000000	100.000000	100.000000	0.000000	0.0	0.000000	0.0
4	ABW	1964	100.000000	100.000000	100.000000	0.000000	0.0	0.000000	0.0
...	...	...	...	...	...	...	...	...	...
16160	ZWE	2016	22.087053	42.561729	85.483994	13.742301	0.0	0.450174	100.0
16161	ZWE	2017	24.531088	44.178635	85.478287	12.394820	0.0	0.405961	100.0
16162	ZWE	2018	26.617121	45.572647	85.468765	11.047340	0.0	0.361748	100.0
16163	ZWE	2019	28.404877	46.781475	85.457336	9.699859	0.0	0.317535	100.0
16164	ZWE	2020	37.060249	52.747669	85.715477	8.352378	0.0	0.273322	100.0

16165 rows × 9 columns

:
```
def dataWideFormat(mainFolderLoc, dataFolderNames):
    data1 = pd.DataFrame()
    for i in range(7):
        data = pd.read_csv(mainFolderLoc+'\\'+dataFolderNames[i]+'\\data.csv')
        data1 = pd.concat([data1,data],ignore_index=True)
    data1.to_csv(mainFolderLoc+'\\dataWideForm.csv', index =False)
```

:
```
dataWideFormat(mainFolderLoc, dataFolderNames)
```