World Bank Data Analysis

This notebook uses the publicly available data at the World Bank websites. Two datasets are used for this analysis,

1. First dataset is the information about countries and its documentation available through World Bank API from the link https://datahelpdesk.worldbank.org/knowledgebase/articles/898590-api-country-queries

1. Second dataset the Gross Domestic Product (GDP) data in CSV format available for download from World Bank Data Catalog available at https://datacatalog.worldbank.org/dataset/global-economic-prospects

In [1]:
```
import requests
import json
from sqlalchemy import create_engine, exc
import pandas as pd
from configparser import ConfigParser
import psycopg2
import urllib
import zipfile
```

In [2]:
```
def get_api_results(endpoint):
    """
      Function to query a api endpoint and converts the results to a json.
      the json data is converted to pandas dataframe before its returned.
      params:
        endpoint (String) - the api endpoint link
      return:
        Dataframe (pandas dataframe) - returns the api results as pandas dataframe.
    """
    response = requests.get(endpoint,timeout=20)
    response = response.json()
    return_lst = response[1]

    while response[0]['page']<response[0]['pages']:
        endpoint_pg = f"{endpoint}&page={int(response[0]['page'])+1}"
        response = requests.get(endpoint_pg,timeout=20)
        response = response.json()
        return_lst = return_lst + response[1]
    return pd.json_normalize(return_lst)
```

In [3]:
```
def get_csv_data(csv_endpoint,filename):
    """
      Function downloads a zip file from the api endpoint. The zip file is extracted
      a local folder. The requested csv file is read and is converted to a pandas da
      params:
        csv_endpoint (String) - the api endpoint link for the zip file
        filename (String)     - csv filename in the format <filename>.csv
      return:
        Dataframe (pandas dataframe) - returns the csv data as a pandas dataframe.
    """
    zip_path, _ = urllib.request.urlretrieve(csv_endpoint)
    with zipfile.ZipFile(zip_path, "r") as f:
        f.extractall("./data")
    return pd.read_csv(f"./data/{filename}")
```

```
# print(get_csv_data("https://databank.worldbank.org/data/download/GEP_CSV.zip","GEP
```

In [4]:

```
def get_db_engine(filename='database.ini', section='postgresql'):
    """
      This function reads database properties file, extracts the postgres db propert
      a database engine using these properties and returns the same.
      params
        filename (String) - database properties file (default value is database.ini)
        section (String) - its the section in the properties file that needs to be r
      return
        database engine  - returns a database engine object.
    """
    parser = ConfigParser()

    # read config file
    parser.read(filename)

    # get section, default to postgresql
    params = {}
    if parser.has_section(section):
        params_ext = parser.items(section)
        for param in params_ext:
            params[param[0]] = param[1]
    else:
        raise Exception('Section {0} not found in the {1} file'.format(section, file

    # intialise the engine variable
    engine = create_engine

    try:

        connect = "postgresql+psycopg2://%s:%s@%s:5432/%s" % (
                    params['user'],
                    params['password'],
                    params['host'],
                    params['database']
                    )
        engine = create_engine(connect)
    except (exc.SQLAlchemyError) as error:
        print(error)

    return engine
```

In [5]:

```
def write_to_db(df,engine,tableName):
    """
      This function writes the pandas dataframe to the database using the db engine
      parameter. The table is replaced evertime its called.
      parameters:
        df (pandas dataframe) : dataframe to be written to the database
        engine(db engine)     : db engine object to write to the database
        tableName(String)     : table name
    """
    df.to_sql(
        tableName,
        con=engine,
        index=False,
        if_exists='replace'
    )
```

In [6]:
```
# Extract the datasets from the API endpoints and conver them to pandas dataframe
worldbank_gdp_df = get_api_results("http://api.worldbank.org/v2/country?format=json"
worldbank_catl_gep_df = get_csv_data("https://databank.worldbank.org/data/download/G

# format the dataframe column names to remove spaces and dots.
worldbank_gdp_df.columns = worldbank_gdp_df.columns.str.replace(".", "", regex=True)
worldbank_catl_gep_df.columns = worldbank_catl_gep_df.columns.str.replace(' ','')

# write the gdp data to the local folder
worldbank_gdp_df.to_csv('./data/worldbankgdp.csv')

# Split the the gdp data into countries and regions.
worldbank_gdp_cnty_df = worldbank_gdp_df.loc[worldbank_gdp_df['regionid']!='NA']
worldbank_gdp_regn_df = worldbank_gdp_df.loc[worldbank_gdp_df['regionid']=='NA']
```

In [7]:
```
# Create a postgres db engine using the properties in the database.ini file
engine = get_db_engine()

# write the dataframes to the postgres database.
write_to_db(worldbank_gdp_cnty_df,engine,'worldBankGdpCnty')
write_to_db(worldbank_gdp_regn_df,engine,'worldBankGdpRegn')
write_to_db(worldbank_catl_gep_df,engine,'worldBankDataCatalogueGep')
```

1. List countries with income level of "Upper middle income"

In [8]:
```
query = """
select "name" as "Country Name","adminregionvalue" as "Region", "incomeLevelvalue" a
from public."worldBankGdpCnty" where "incomeLevelid" = 'UMC'
"""

df = pd.read_sql(query,engine)
df
```

Out[8]:

	Country Name	Region	Income Level
0	Albania	Europe & Central Asia (excluding high income)	Upper middle income
1	Argentina	Latin America & Caribbean (excluding high income)	Upper middle income
2	Armenia	Europe & Central Asia (excluding high income)	Upper middle income
3	American Samoa	East Asia & Pacific (excluding high income)	Upper middle income
4	Azerbaijan	Europe & Central Asia (excluding high income)	Upper middle income
5	Bulgaria	Europe & Central Asia (excluding high income)	Upper middle income
6	Bosnia and Herzegovina	Europe & Central Asia (excluding high income)	Upper middle income
7	Belarus	Europe & Central Asia (excluding high income)	Upper middle income
8	Brazil	Latin America & Caribbean (excluding high income)	Upper middle income

	Country Name	Region	Income Level
9	Botswana	Sub-Saharan Africa (excluding high income)	Upper middle income
10	China	East Asia & Pacific (excluding high income)	Upper middle income
11	Colombia	Latin America & Caribbean (excluding high income)	Upper middle income
12	Costa Rica	Latin America & Caribbean (excluding high income)	Upper middle income
13	Cuba	Latin America & Caribbean (excluding high income)	Upper middle income
14	Dominica	Latin America & Caribbean (excluding high income)	Upper middle income
15	Dominican Republic	Latin America & Caribbean (excluding high income)	Upper middle income
16	Ecuador	Latin America & Caribbean (excluding high income)	Upper middle income
17	Fiji	East Asia & Pacific (excluding high income)	Upper middle income
18	Gabon	Sub-Saharan Africa (excluding high income)	Upper middle income
19	Georgia	Europe & Central Asia (excluding high income)	Upper middle income
20	Equatorial Guinea	Sub-Saharan Africa (excluding high income)	Upper middle income
21	Grenada	Latin America & Caribbean (excluding high income)	Upper middle income
22	Guatemala	Latin America & Caribbean (excluding high income)	Upper middle income
23	Guyana	Latin America & Caribbean (excluding high income)	Upper middle income
24	Iraq	Middle East & North Africa (excluding high inc...	Upper middle income
25	Jamaica	Latin America & Caribbean (excluding high income)	Upper middle income
26	Jordan	Middle East & North Africa (excluding high inc...	Upper middle income
27	Kazakhstan	Europe & Central Asia (excluding high income)	Upper middle income
28	Lebanon	Middle East & North Africa (excluding high inc...	Upper middle income
29	Libya	Middle East & North Africa (excluding high inc...	Upper middle income
30	St. Lucia	Latin America & Caribbean (excluding high income)	Upper middle income
31	Moldova	Europe & Central Asia (excluding high income)	Upper middle income

	Country Name	Region	Income Level
32	Maldives	South Asia	Upper middle income
33	Mexico	Latin America & Caribbean (excluding high income)	Upper middle income
34	Marshall Islands	East Asia & Pacific (excluding high income)	Upper middle income
35	North Macedonia	Europe & Central Asia (excluding high income)	Upper middle income
36	Montenegro	Europe & Central Asia (excluding high income)	Upper middle income
37	Mauritius	Sub-Saharan Africa (excluding high income)	Upper middle income
38	Malaysia	East Asia & Pacific (excluding high income)	Upper middle income
39	Namibia	Sub-Saharan Africa (excluding high income)	Upper middle income
40	Panama	Latin America & Caribbean (excluding high income)	Upper middle income
41	Peru	Latin America & Caribbean (excluding high income)	Upper middle income
42	Paraguay	Latin America & Caribbean (excluding high income)	Upper middle income
43	Romania	Europe & Central Asia (excluding high income)	Upper middle income
44	Russian Federation	Europe & Central Asia (excluding high income)	Upper middle income
45	Serbia	Europe & Central Asia (excluding high income)	Upper middle income
46	Suriname	Latin America & Caribbean (excluding high income)	Upper middle income
47	Thailand	East Asia & Pacific (excluding high income)	Upper middle income
48	Turkmenistan	Europe & Central Asia (excluding high income)	Upper middle income
49	Tonga	East Asia & Pacific (excluding high income)	Upper middle income
50	Turkey	Europe & Central Asia (excluding high income)	Upper middle income
51	Tuvalu	East Asia & Pacific (excluding high income)	Upper middle income
52	St. Vincent and the Grenadines	Latin America & Caribbean (excluding high income)	Upper middle income
53	Kosovo	Europe & Central Asia (excluding high income)	Upper middle income
54	South Africa	Sub-Saharan Africa (excluding high income)	Upper middle income

2. List countries with income level of "Low income" per region.

In [9]:
```
query = """
select "regionid", "adminregionvalue" as "Region", "name" as "Country Name",
"incomeLevelvalue" as "Income Level"
from public."worldBankGdpCnty"
where "incomeLevelid" = 'LIC'
group by (regionid, adminregionvalue,"name", "incomeLevelvalue")
order by regionid
"""

df = pd.read_sql(query,engine)
df
```

Out[9]:

	regionid	Region	Country Name	Income Level
0	EAS	East Asia & Pacific (excluding high income)	Korea, Dem. People's Rep.	Low income
1	MEA	Middle East & North Africa (excluding high inc...	Syrian Arab Republic	Low income
2	MEA	Middle East & North Africa (excluding high inc...	Yemen, Rep.	Low income
3	SAS	South Asia	Afghanistan	Low income
4	SSF	Sub-Saharan Africa (excluding high income)	Burkina Faso	Low income
5	SSF	Sub-Saharan Africa (excluding high income)	Burundi	Low income
6	SSF	Sub-Saharan Africa (excluding high income)	Central African Republic	Low income
7	SSF	Sub-Saharan Africa (excluding high income)	Chad	Low income
8	SSF	Sub-Saharan Africa (excluding high income)	Congo, Dem. Rep.	Low income
9	SSF	Sub-Saharan Africa (excluding high income)	Eritrea	Low income
10	SSF	Sub-Saharan Africa (excluding high income)	Ethiopia	Low income
11	SSF	Sub-Saharan Africa (excluding high income)	Gambia, The	Low income
12	SSF	Sub-Saharan Africa (excluding high income)	Guinea	Low income
13	SSF	Sub-Saharan Africa (excluding high income)	Guinea-Bissau	Low income
14	SSF	Sub-Saharan Africa (excluding high income)	Liberia	Low income
15	SSF	Sub-Saharan Africa (excluding high income)	Madagascar	Low income
16	SSF	Sub-Saharan Africa (excluding high income)	Malawi	Low income
17	SSF	Sub-Saharan Africa (excluding high income)	Mali	Low income
18	SSF	Sub-Saharan Africa (excluding high income)	Mozambique	Low income
19	SSF	Sub-Saharan Africa (excluding high income)	Niger	Low income
20	SSF	Sub-Saharan Africa (excluding high income)	Rwanda	Low income
21	SSF	Sub-Saharan Africa (excluding high income)	Sierra Leone	Low income
22	SSF	Sub-Saharan Africa (excluding high income)	Somalia	Low income
23	SSF	Sub-Saharan Africa (excluding high income)	South Sudan	Low income
24	SSF	Sub-Saharan Africa (excluding high income)	Sudan	Low income
25	SSF	Sub-Saharan Africa (excluding high income)	Togo	Low income

	regionid	Region	Country Name	Income Level
26	SSF	Sub-Saharan Africa (excluding high income)	Uganda	Low income

3. Find the region with the highest proportion of "High income" countries.

In [10]:
```
query = """
select rslt.name as "Region",rslt.noofcountries "No of countries in High Income leve
from(
select cnty.regionid,cnty.noofcountries, RANK() OVER( ORDER BY cnty.noofcountries DE
from (
select regionid,count(name) as noofcountries
from public."worldBankGdpCnty"
where "incomeLevelid" = 'HIC'
group by (regionid)) cnty
LEFT JOIN public."worldBankGdpRegn" as reg on reg.id = cnty.regionid ) rslt
where rslt.region_rank = 1;
"""

df = pd.read_sql(query,engine)
df
```

Out[10]:

	Region	No of countries in High Income level	Region Rank
0	Europe & Central Asia	37	1

4. Calculate cumulative/running value of GDP per region ordered by income from lowest to highest and country

Income value is not provided in any of the tables so query with order by income is not possible. The query could be updated to have order by cumulative GDP for a year. below query shows the regions with ascending cumulative GDP for year 2021.

In [11]:
```
query = """
select final_rslt.regionvalue, final_rslt.sum_2021
from (
select cnty.regionid, cnty.regionvalue,
ROUND (sum(rslt."2018")::numeric,2) as sum_2018,
ROUND (sum(rslt."2019")::numeric,2) as sum_2019,
ROUND (sum(rslt."2020")::numeric,2) as sum_2020,
ROUND (sum(rslt."2021")::numeric,2) as sum_2021,
ROUND (sum(rslt."2022")::numeric,2) as sum_2022
from
(select "CountryCode","2018","2019","2020","2021","2022"
from public."worldBankDataCatalogueGep"
where "CountryCode" <> ALL (array['AME','EAA','EMD','E19','ECH','LAP','MNH','SAP','S
LEFT JOIN public."worldBankGdpCnty" as cnty on cnty.id = rslt."CountryCode"
GROUP BY (cnty.regionid,cnty.regionvalue)) final_rslt
GROUP BY (final_rslt.regionvalue,final_rslt.sum_2021)
ORDER BY (sum_2021)
"""

df = pd.read_sql(query,engine)
df
```

Out[11]:

	regionvalue	sum_2021

	regionvalue	sum_2021
0	North America	3.5
1	South Asia	22.7
2	Middle East & North Africa	27.0
3	East Asia & Pacific	65.7
4	Europe & Central Asia	78.2
5	Latin America & Caribbean	96.3
6	Sub-Saharan Africa	134.3

5 Calculate percentage difference in value of GDP year-on-year per country

In [12]:

```
query = """
select "CountryName", "CountryCode",
(("2019"-"2018")*100)/(CASE WHEN "2018"=0 THEN 1 ELSE "2018" END) as "YoYPctChange20
(("2020"-"2019")*100)/(CASE WHEN "2019"=0 THEN 1 ELSE "2019" END) as "YoYPctChange20
(("2021"-"2020")*100)/(CASE WHEN "2020"=0 THEN 1 ELSE "2020" END) as "YoYPctChange20
(("2022"-"2021")*100)/(CASE WHEN "2021"=0 THEN 1 ELSE "2021" END) as "YoYPctChange20
from
(select "CountryName", "CountryCode", "IndicatorName","IndicatorCode","2018","2019",
from public."worldBankDataCatalogueGep"
where "CountryCode" <> ALL (array['AME','EAA','EMD','E19','ECH','LAP','MNH','SAP','S
"""

df = pd.read_sql(query,engine)
df
```

Out[12]:

	CountryName	CountryCode	YoYPctChange2019	YoYPctChange2020	YoYPctChange2021	YoYP
0	Afghanistan	AFG	225.000000	-241.025641	-145.454545	
1	Albania	ALB	-46.341463	-404.545455	-176.119403	
2	Algeria	DZA	-33.333333	-912.500000	-158.461538	
3	Angola	AGO	-55.000000	344.444444	-122.500000	
4	Argentina	ARG	-19.230769	404.761905	-146.226415	
...	...	...	...	...	...	
131	Vietnam	VNM	-1.408451	-60.000000	139.285714	
132	West Bank and Gaza	PSE	16.666667	-664.285714	-129.113924	
133	Zambia	ZMB	-60.000000	-421.428571	-142.222222	
134	Zimbabwe	ZWE	-268.750000	23.456790	-129.000000	
135	South Sudan	SSD	-91.428571	-3200.000000	-136.559140	

136 rows × 6 columns

6 List 3 countries with lowest GDP per region.

In [13]:
```
query = """
select finalrslt.*
from(
select cnty.regionid, cnty.regionvalue, cnty.id,
rslt."CountryName", rslt."2021" as "gdp2021",
RANK() OVER(PARTITION BY cnty.regionid ORDER BY rslt."2021" ASC) as "country_rank_wi
from
(select "CountryName", "CountryCode", "IndicatorName","IndicatorCode","2018","2019",
from public."worldBankDataCatalogueGep"
where "CountryCode" <> ALL (array['AME','EAA','EMD','E19','ECH','LAP','MNH','SAP','S
LEFT JOIN public."worldBankGdpCnty" as cnty on cnty.id = rslt."CountryCode" ) finalr
WHERE finalrslt.country_rank_within_region < 4
ORDER BY finalrslt.regionid,finalrslt.country_rank_within_region
"""

df = pd.read_sql(query,engine)
df
```

Out[13]:

	regionid	regionvalue	id	CountryName	gdp2021	country_rank_within_region
0	EAS	East Asia & Pacific	MMR	Myanmar	2.0	1
1	EAS	East Asia & Pacific	JPN	Japan	2.5	2
2	EAS	East Asia & Pacific	FJI	Fiji	2.6	3
3	ECS	Europe & Central Asia	BLR	Belarus	-2.7	1
4	ECS	Europe & Central Asia	AZE	Azerbaijan	1.9	2
5	ECS	Europe & Central Asia	KAZ	Kazakhstan	2.5	3
6	LCN	Latin America & Caribbean	SUR	Suriname	-1.9	1
7	LCN	Latin America & Caribbean	NIC	Nicaragua	-0.9	2
8	LCN	Latin America & Caribbean	VCT	St. Vincent and the Grenadines	0.0	3
9	MEA	Middle East & North Africa	LBN	Lebanon	-13.2	1
10	MEA	Middle East & North Africa	KWT	Kuwait	0.5	2
11	MEA	Middle East & North Africa	OMN	Oman	0.5	2
12	NAC	North America	USA	United States	3.5	1
13	SAS	South Asia	BTN	Bhutan	-0.7	1
14	SAS	South Asia	PAK	Pakistan	0.5	2
15	SAS	South Asia	NPL	Nepal	0.6	3
16	SSF	Sub-Saharan Africa	SSD	South Sudan	-3.4	1
17	SSF	Sub-Saharan Africa	GNQ	Equatorial Guinea	-2.8	2

	regionid	regionvalue	id	CountryName	gdp2021	country_rank_within_region
18	SSF	Sub-Saharan Africa	COG	Congo, Rep.	-2.0	3

7. Provide an interesting fact from the dataset.

In [ ]:

```
# tbd
```