Window/Analytical Functions in SQL

Team 1

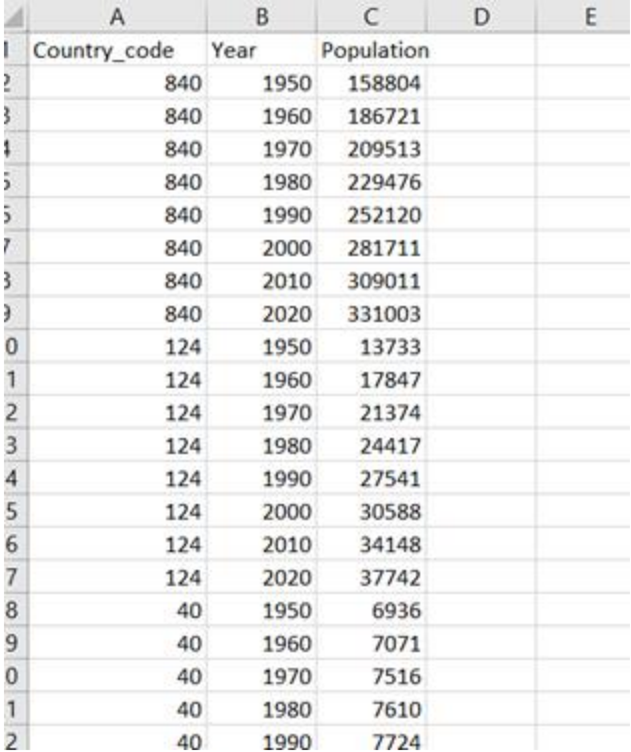
Abhishek
Parveen
Chandana
Shruthi

October 24, 2020

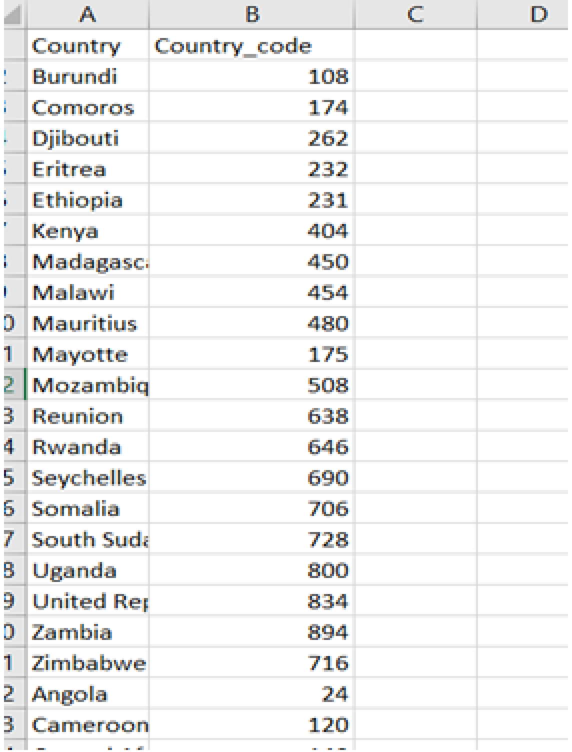
Project Objective:

Analyse the country wise population over the decades since the year 1950

My Data :
World population country wise change over decade



A	B	C	D	E
Country_code	Year	Population		
840	1950	158804		
840	1960	186721		
840	1970	209513		
840	1980	229476		
840	1990	252120		
840	2000	281711		
840	2010	309011		
840	2020	331003		
124	1950	13733		
124	1960	17847		
124	1970	21374		
124	1980	24417		
124	1990	27541		
124	2000	30588		
124	2010	34148		
124	2020	37742		
40	1950	6936		
40	1960	7071		
40	1970	7516		
40	1980	7610		
40	1990	7724		



A	B	C	D
Country	Country_code		
Burundi	108		
Comoros	174		
Djibouti	262		
Eritrea	232		
Ethiopia	231		
Kenya	404		
Madagasc	450		
Malawi	454		
Mauritius	480		
Mayotte	175		
Mozambiq	508		
Reunion	638		
Rwanda	646		
Seychelles	690		
Somalia	706		
South Suda	728		
Uganda	800		
United Rep	834		
Zambia	894		
Zimbabwe	716		
Angola	24		
Cameroon	120		

Reference:
https://population.un.org/wpp/Download/

Note : Only few countries population with change over decades from 1950 is taken for the analysis

What functions are used for this analysis?

1. cume_dist()
2. dense_rank()
3. lead()
4. Python and SQL Connectivity with any Dataset using any AGGREGATE/Analytic Window Query.

Function: cume_dist()

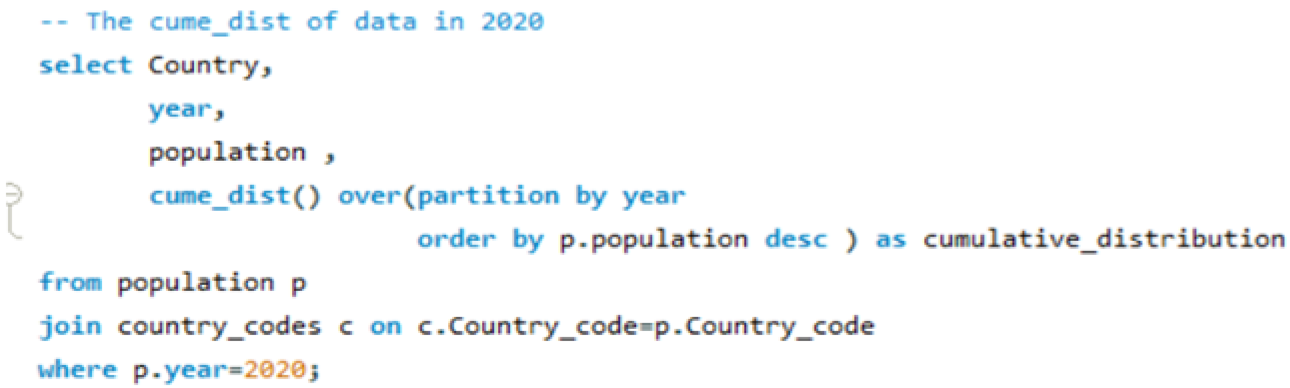
The **CUME_DIST()** is a window function that returns the cumulative distribution of a value within a set of values. It represents the number of rows with values less than or equal to that row’s value divided by the total number of rows.

The following shows the syntax of the **CUME_DIST()** function:

CUME_DIST() OVER (

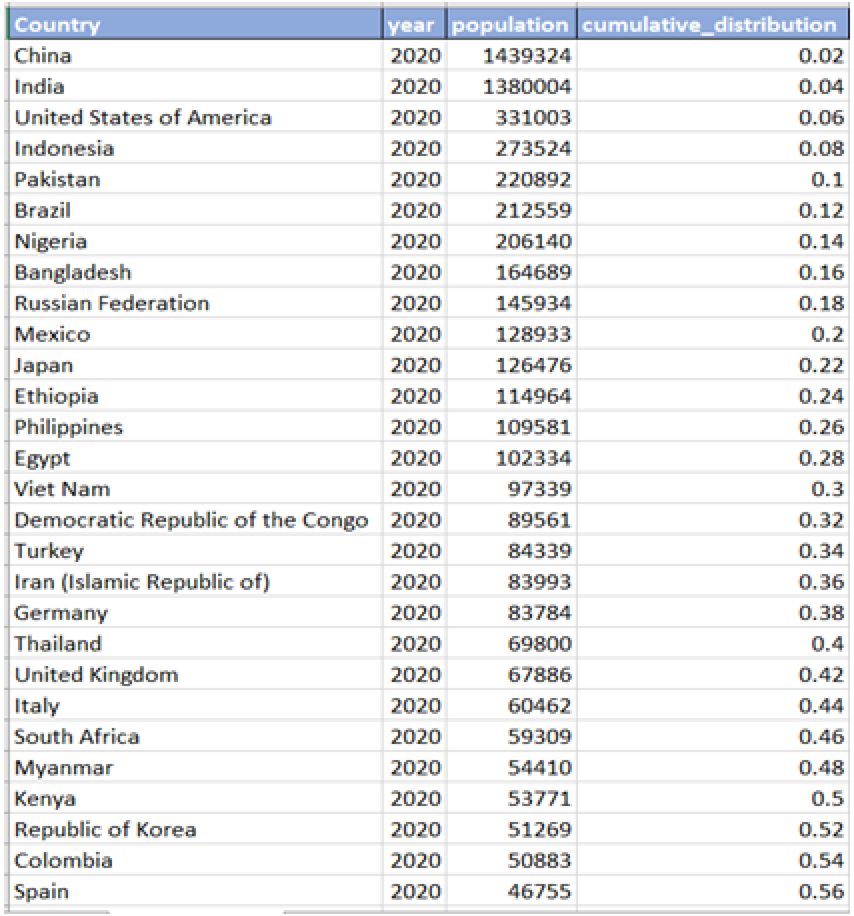
PARTITION BY expr, ...

ORDER BY expr [ASC | DESC], ...)



```
-- The cume_dist of data in 2020
select Country,
       year,
       population ,
       cume_dist() over(partition by year
                        order by p.population desc ) as cumulative_distribution
from population p
join country_codes c on c.Country_code=p.Country_code
where p.year=2020;
```

Inferences from function cume_dist()



Country	year	population	cumulative_distribution
China	2020	1439324	0.02
India	2020	1380004	0.04
United States of America	2020	331003	0.06
Indonesia	2020	273524	0.08
Pakistan	2020	220892	0.1
Brazil	2020	212559	0.12
Nigeria	2020	206140	0.14
Bangladesh	2020	164689	0.16
Russian Federation	2020	145934	0.18
Mexico	2020	128933	0.2
Japan	2020	126476	0.22
Ethiopia	2020	114964	0.24
Philippines	2020	109581	0.26
Egypt	2020	102334	0.28
Viet Nam	2020	97339	0.3
Democratic Republic of the Congo	2020	89561	0.32
Turkey	2020	84339	0.34
Iran (Islamic Republic of)	2020	83993	0.36
Germany	2020	83784	0.38
Thailand	2020	69800	0.4
United Kingdom	2020	67886	0.42
Italy	2020	60462	0.44
South Africa	2020	59309	0.46
Myanmar	2020	54410	0.48
Kenya	2020	53771	0.5
Republic of Korea	2020	51269	0.52
Colombia	2020	50883	0.54
Spain	2020	46755	0.56

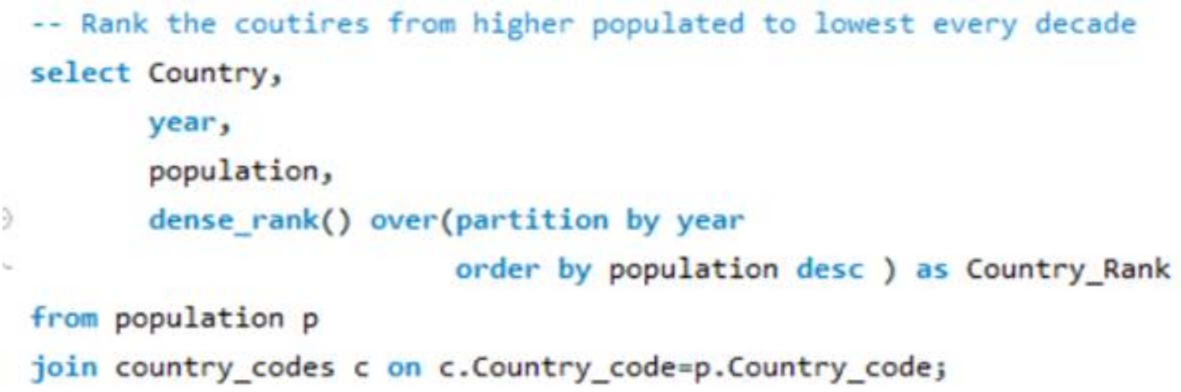
1. 50% of the world population is above 53771 in the year 2020.
2. We can also infer the population distribution percentages comparing multiple years and for the required countries.
3. This helps in finding out the countries which are above or below the threshold populations and by what percent.

Function: dense_rank()

The **DENSE_RANK()** is a window function that assigns a rank to each row within a partition or result set with no gaps in ranking values.The rank of a row is increased by one from the number of distinct rank values which come before the row.

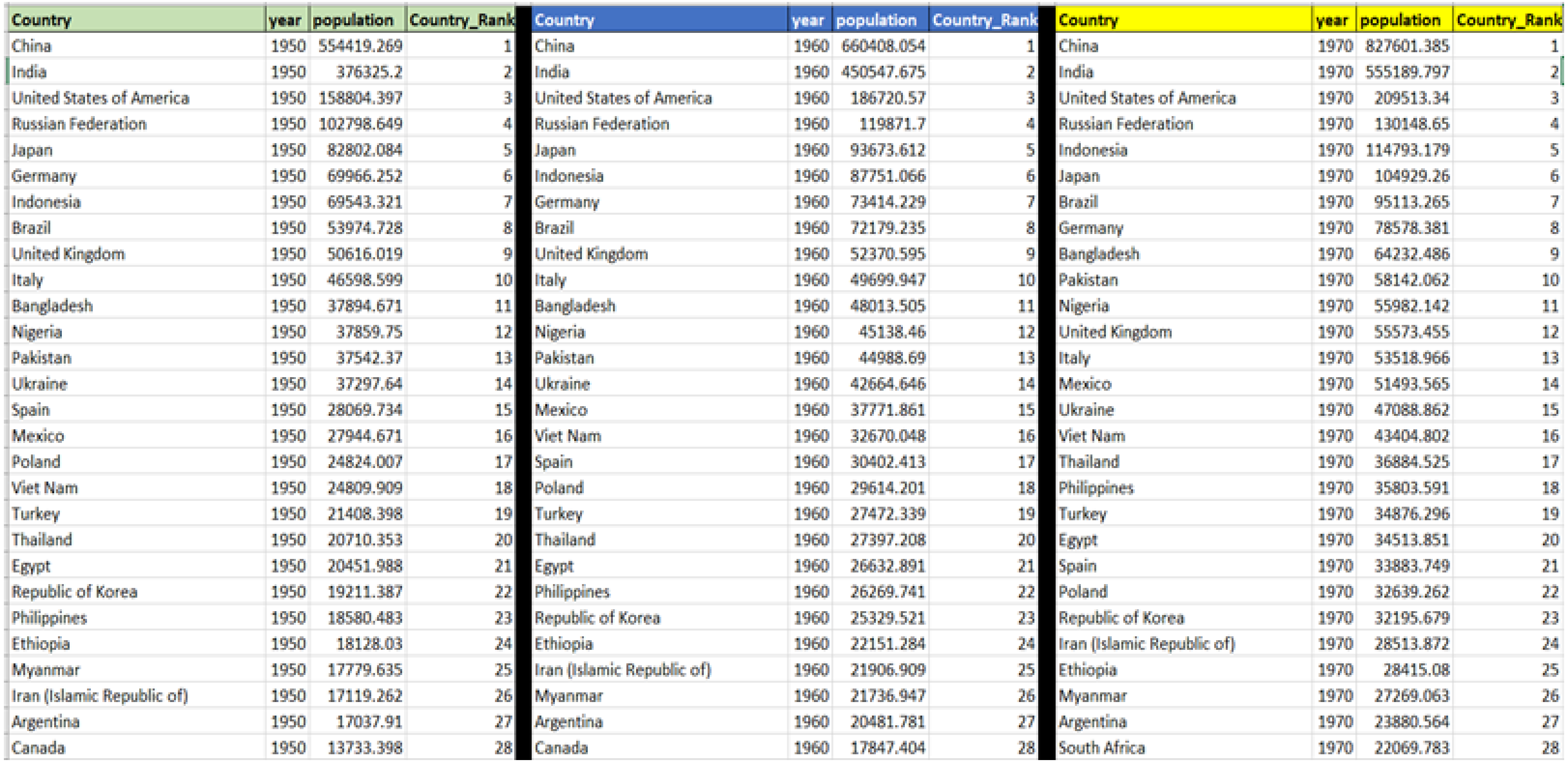
The syntax of the **DENSE_RANK()** function is as follows:

```
DENSE_RANK() OVER (

  PARTITION BY <expression>[{,<expression>...}]

  ORDER BY <expression> [ASC|DESC], [{,<expression>...}])
```



```
-- Rank the coutires from higher populated to lowest every decade
select Country,
       year,
       population,
       dense_rank() over(partition by year
                         order by population desc ) as Country_Rank
from population p
join country_codes c on c.Country_code=p.Country_code;
```

Where do the countries stand based on population?



Country	year	population	Country_Rank
China	1950	554419.269	1
India	1950	376325.2	2
United States of America	1950	158804.397	3
Russian Federation	1950	102798.649	4
Japan	1950	82802.084	5
Germany	1950	69966.252	6
Indonesia	1950	69543.321	7
Brazil	1950	53974.728	8
United Kingdom	1950	50616.019	9
Italy	1950	46598.599	10
Bangladesh	1950	37894.671	11
Nigeria	1950	37859.75	12
Pakistan	1950	37542.37	13
Ukraine	1950	37297.64	14
Spain	1950	28069.734	15
Mexico	1950	27944.671	16
Poland	1950	24824.007	17
Viet Nam	1950	24809.909	18
Turkey	1950	21408.398	19
Thailand	1950	20710.353	20
Egypt	1950	20451.988	21
Republic of Korea	1950	19211.387	22
Philippines	1950	18580.483	23
Ethiopia	1950	18128.03	24
Myanmar	1950	17779.635	25
Iran (Islamic Republic of)	1950	17119.262	26
Argentina	1950	17037.91	27
Canada	1950	13733.398	28

Country	year	population	Country_Rank
China	1960	660408.054	1
India	1960	450547.675	2
United States of America	1960	186720.57	3
Russian Federation	1960	119871.7	4
Japan	1960	93673.612	5
Indonesia	1960	87751.066	6
Germany	1960	73414.229	7
Brazil	1960	72179.235	8
United Kingdom	1960	52370.595	9
Italy	1960	49699.947	10
Bangladesh	1960	48013.505	11
Nigeria	1960	45138.46	12
Pakistan	1960	44988.69	13
Ukraine	1960	42664.646	14
Mexico	1960	37771.861	15
Viet Nam	1960	32670.048	16
Spain	1960	30402.413	17
Poland	1960	29614.201	18
Turkey	1960	27472.339	19
Thailand	1960	27397.208	20
Egypt	1960	26632.891	21
Philippines	1960	26269.741	22
Republic of Korea	1960	25329.521	23
Ethiopia	1960	22151.284	24
Iran (Islamic Republic of)	1960	21906.909	25
Myanmar	1960	21736.947	26
Argentina	1960	20481.781	27
Canada	1960	17847.404	28

Country	year	population	Country_Rank
China	1970	827601.385	1
India	1970	555189.797	2
United States of America	1970	209513.34	3
Russian Federation	1970	130148.65	4
Indonesia	1970	114793.179	5
Japan	1970	104929.26	6
Brazil	1970	95113.265	7
Germany	1970	78578.381	8
Bangladesh	1970	64232.486	9
Pakistan	1970	58142.062	10
Nigeria	1970	55982.142	11
United Kingdom	1970	55573.455	12
Italy	1970	53518.966	13
Mexico	1970	51493.565	14
Ukraine	1970	47088.862	15
Viet Nam	1970	43404.802	16
Thailand	1970	36884.525	17
Philippines	1970	35803.591	18
Turkey	1970	34876.296	19
Egypt	1970	34513.851	20
Spain	1970	33883.749	21
Poland	1970	32639.262	22
Republic of Korea	1970	32195.679	23
Iran (Islamic Republic of)	1970	28513.872	24
Ethiopia	1970	28415.08	25
Myanmar	1970	27269.063	26
Argentina	1970	23880.564	27
South Africa	1970	22069.783	28

It is observed that many countries has fluctuations in their population in between 10 years. Some has got up higher in rank as well as some lowered in their population rank.

Function: rank()-

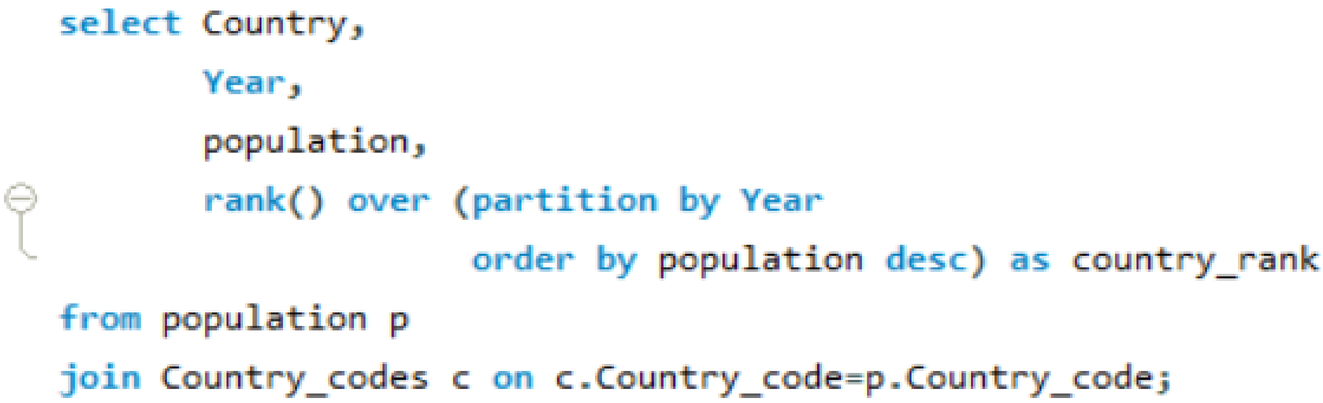
The **RANK()** function assigns a rank to each row within the partition of a result set. The rank of a row is specified by one plus the number of ranks that come before it.

The following shows the syntax of the **RANK()** function:

RANK() OVER (

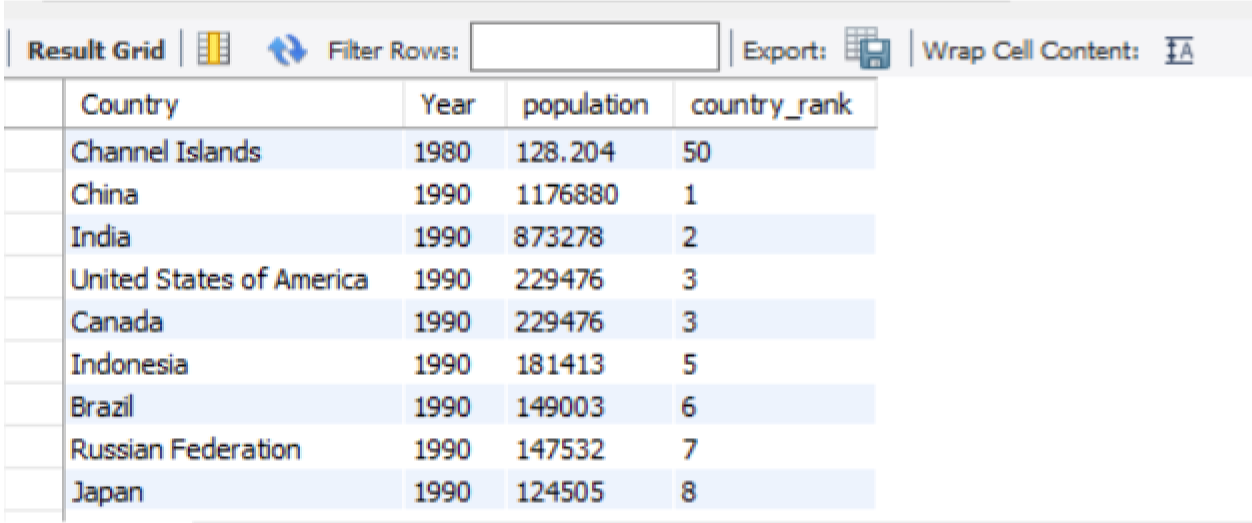
PARTITION BY <expression>[{,<expression>...}]

ORDER BY <expression> [ASC|DESC], [{,<expression>...}])



```
select Country,
       Year,
       population,
       rank() over (partition by Year
                    order by population desc) as country_rank
from population p
join Country_codes c on c.Country_code=p.Country_code;
```

Function: rank()- Contd.



Country	Year	population	country_rank
Channel Islands	1980	128.204	50
China	1990	1176880	1
India	1990	873278	2
United States of America	1990	229476	3
Canada	1990	229476	3
Indonesia	1990	181413	5
Brazil	1990	149003	6
Russian Federation	1990	147532	7
Japan	1990	124505	8

The only difference between the RANK() and DENSE_RANK() functions is in cases where there is a "tie"; i.e., in cases where multiple values in a set have the same ranking. In such cases, RANK() will assign non-consecutive "ranks" to the values in the set (resulting in gaps between the integer ranking values when there is a tie), whereas DENSE_RANK() will assign consecutive ranks to the values in the set (so there will be no gaps between the integer ranking values in the case of a tie).

Function: lead()

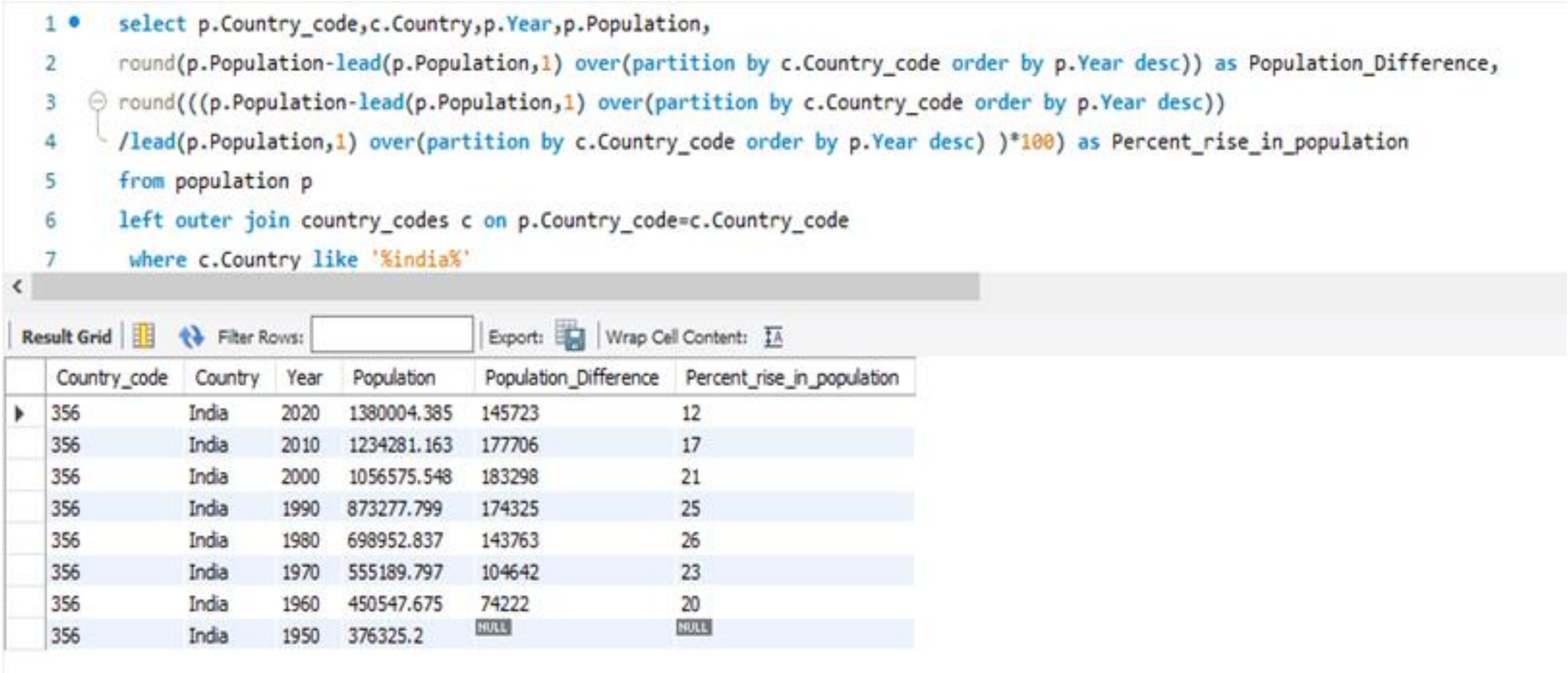
The **LEAD()** function is a window function that allows you to look forward a number of rows and access data of that row from the current row.The **LEAD()** function is very useful for calculating the difference between the current row and the subsequent row within the same result set.

The following shows the syntax of the **LEAD()** function:

```
LEAD(<expression>[,offset[, default_value]]) OVER (
    PARTITION BY (expr)
    ORDER BY (expr))
```

Function: lead() Contd.

India’s Population Growth in last 8 decades. It is observed that the higher percentage of population rise was between the years 1970 and 1990

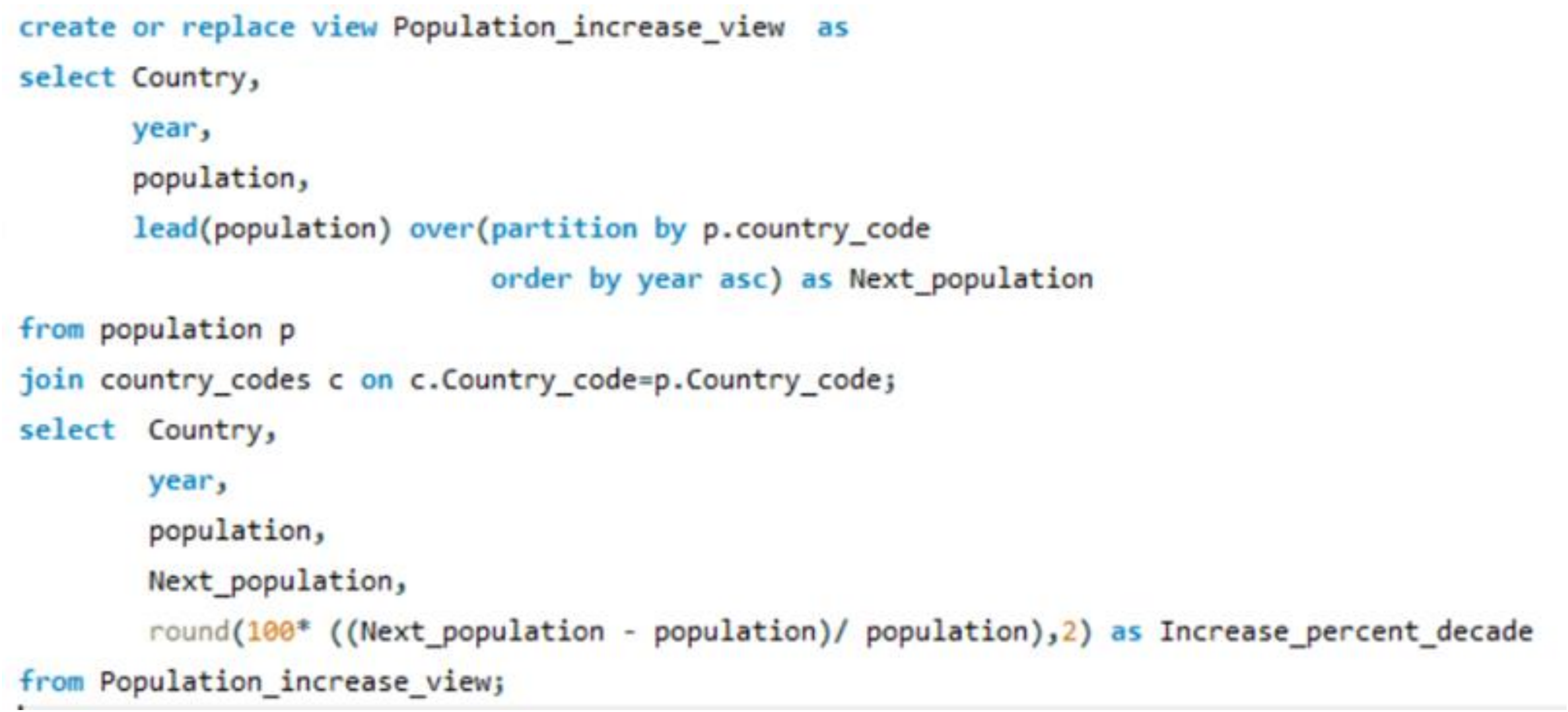


```
select p.Country_code,c.Country,p.Year,p.Population,
round(p.Population-lead(p.Population,1) over(partition by c.Country_code order by p.Year desc)) as Population_Difference,
round(((p.Population-lead(p.Population,1) over(partition by c.Country_code order by p.Year desc))
/lead(p.Population,1) over(partition by c.Country_code order by p.Year desc) )*100) as Percent_rise_in_population
from population p
left outer join country_codes c on p.Country_code=c.Country_code
 where c.Country like '%india%'
```

Country_code	Country	Year	Population	Population_Difference	Percent_rise_in_population
356	India	2020	1380004.385	145723	12
356	India	2010	1234281.163	177706	17
356	India	2000	1056575.548	183298	21
356	India	1990	873277.799	174325	25
356	India	1980	698952.837	143763	26
356	India	1970	555189.797	104642	23
356	India	1960	450547.675	74222	20
356	India	1950	376325.2	NULL	NULL

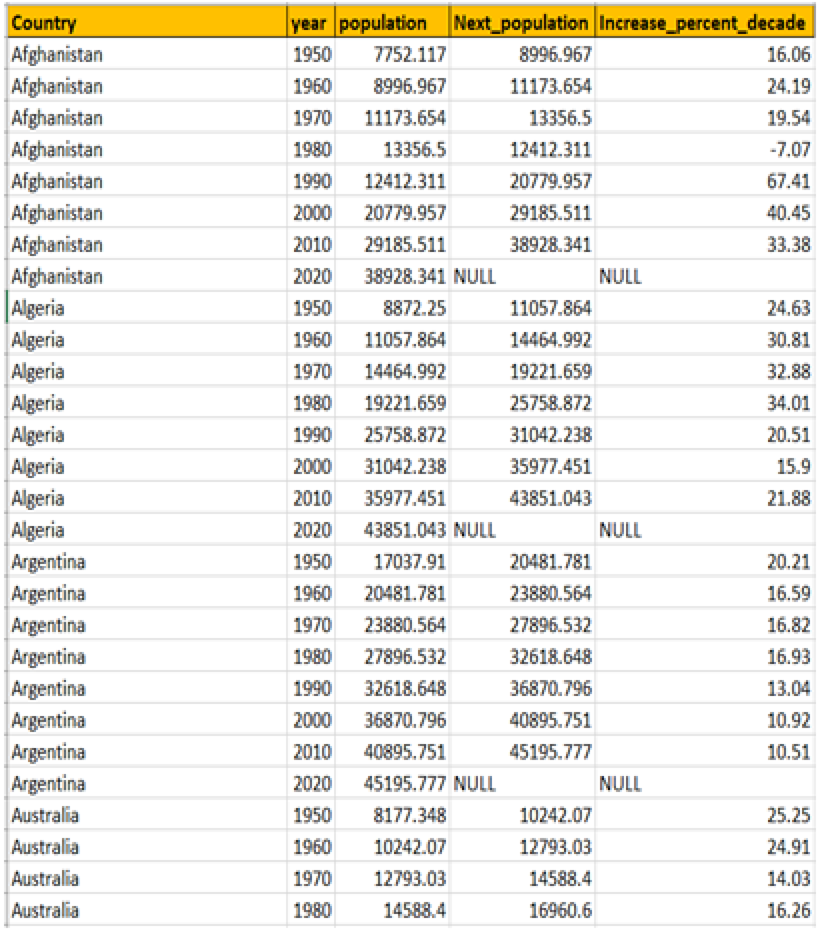
Function: lead() Contd.

Check the population increase in 10 years



```
create or replace view Population_increase_view  as
select Country,
       year,
       population,
       lead(population) over(partition by p.country_code
                             order by year asc) as Next_population
from population p
join country_codes c on c.Country_code=p.Country_code;
select  Country,
        year,
        population,
        Next_population,
        round(100* ((Next_population - population)/ population),2) as Increase_percent_decade
from Population_increase_view;
```

Function: lead() Contd.



Country	year	population	Next_population	Increase_percent_decade
Afghanistan	1950	7752.117	8996.967	16.06
Afghanistan	1960	8996.967	11173.654	24.19
Afghanistan	1970	11173.654	13356.5	19.54
Afghanistan	1980	13356.5	12412.311	-7.07
Afghanistan	1990	12412.311	20779.957	67.41
Afghanistan	2000	20779.957	29185.511	40.45
Afghanistan	2010	29185.511	38928.341	33.38
Afghanistan	2020	38928.341	NULL	NULL
Algeria	1950	8872.25	11057.864	24.63
Algeria	1960	11057.864	14464.992	30.81
Algeria	1970	14464.992	19221.659	32.88
Algeria	1980	19221.659	25758.872	34.01
Algeria	1990	25758.872	31042.238	20.51
Algeria	2000	31042.238	35977.451	15.9
Algeria	2010	35977.451	43851.043	21.88
Algeria	2020	43851.043	NULL	NULL
Argentina	1950	17037.91	20481.781	20.21
Argentina	1960	20481.781	23880.564	16.59
Argentina	1970	23880.564	27896.532	16.82
Argentina	1980	27896.532	32618.648	16.93
Argentina	1990	32618.648	36870.796	13.04
Argentina	2000	36870.796	40895.751	10.92
Argentina	2010	40895.751	45195.777	10.51
Argentina	2020	45195.777	NULL	NULL
Australia	1950	8177.348	10242.07	25.25
Australia	1960	10242.07	12793.03	24.91
Australia	1970	12793.03	14588.4	14.03
Australia	1980	14588.4	16960.6	16.26

A generic trend of rise and fall in the population can be observed for each country over the years.

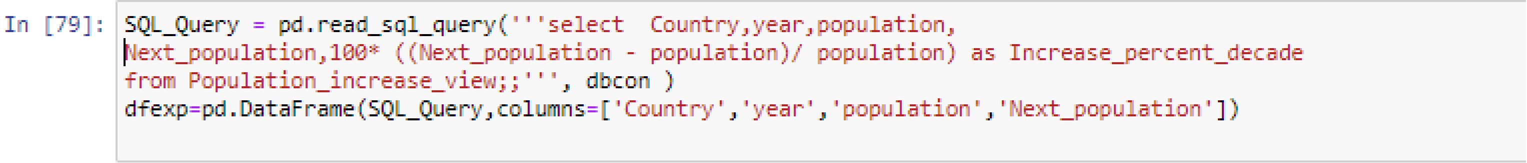
Python and SQL Connectivity with any Dataset using any AGGREGATE/Analytic Window Query

The world's **human population is growing at an exponential rate**. Let us see how the growth rate over the years for different countries?.

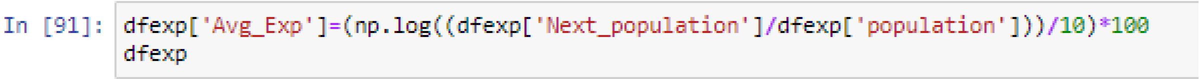
Average exponential rate of growth of the population over a given period. It is calculated as ln(Pt/P0)/t where t is the length of the period. It is expressed as a percentage

pymysql for connectivity

dbcon = pymysql.connect("localhost", "root", "password","world")



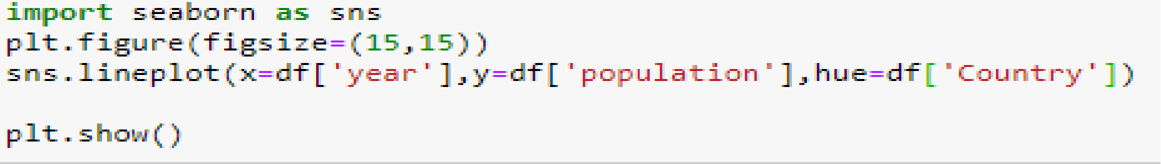
```
In [79]: SQL_Query = pd.read_sql_query('''select  Country,year,population,
         Next_population,100* ((Next_population - population)/ population) as Increase_percent_decade
         from Population_increase_view;;''', dbcon )
         dfexp=pd.DataFrame(SQL_Query,columns=['Country','year','population','Next_population'])
```



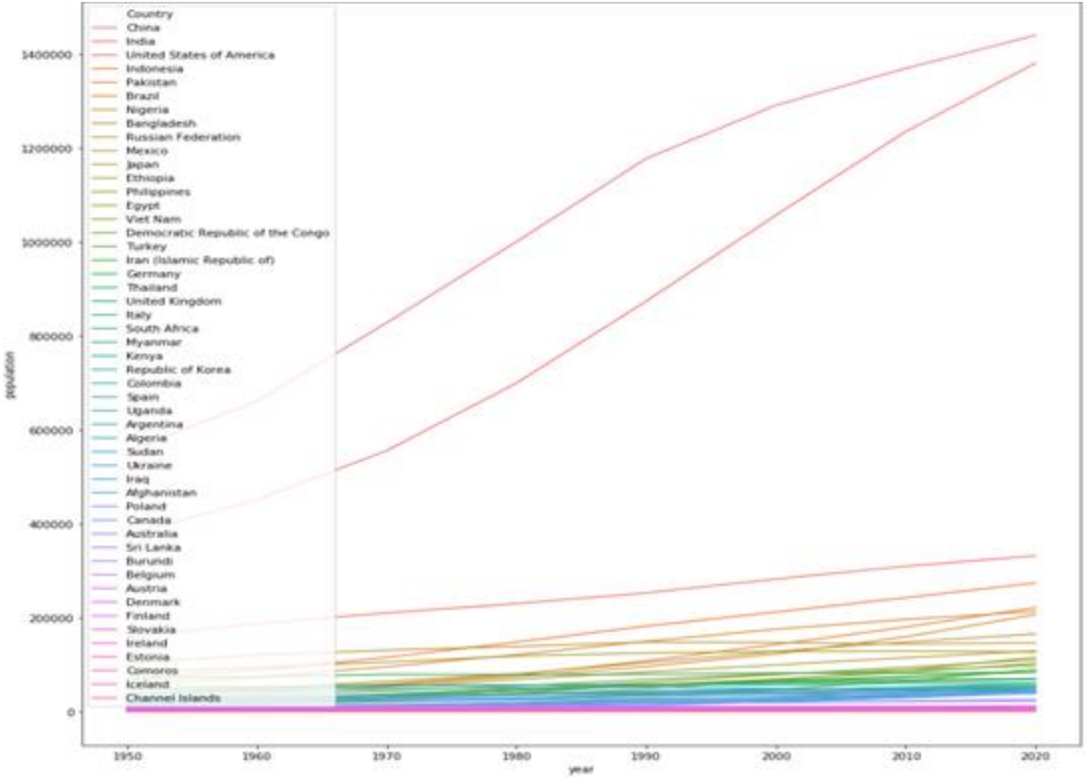
```
In [91]: dfexp['Avg_Exp']=(np.log((dfexp['Next_population']/dfexp['population']))/10)*100
         dfexp
```

Note: LEAD is an **analytic function which is used in Population_increase_view in *sql file.**

Graphical Inferences from population data



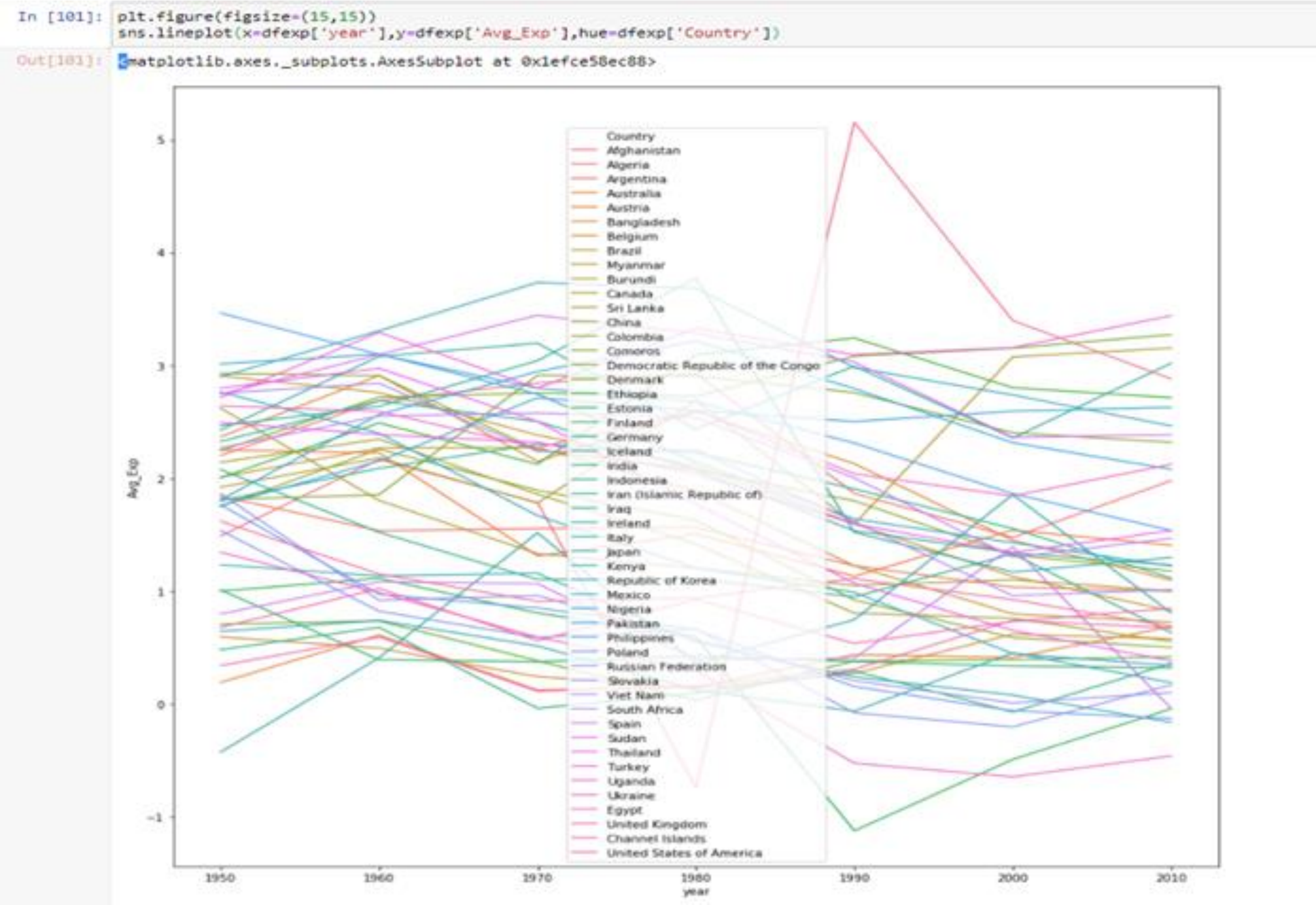
```
import seaborn as sns
plt.figure(figsize=(15,15))
sns.lineplot(x=df['year'],y=df['population'],hue=df['Country'])

plt.show()
```



Population growth graph of countries since 1950 in steps of 10 years

Which countries are ahead in the population growth?

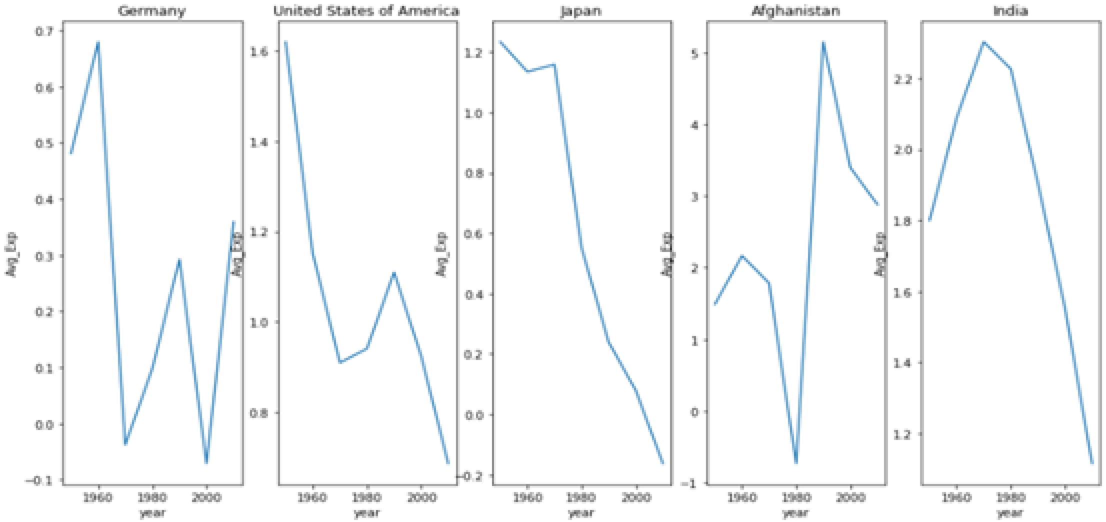
Graphical Inferences from population data contd.



```
In [101]: plt.figure(figsize=(15,15))
          sns.lineplot(x=dfexp['year'],y=dfexp['Avg_Exp'],hue=dfexp['Country'])

Out[101]: <matplotlib.axes._subplots.AxesSubplot at 0x1efce58ec88>
```

Graphical Inferences from population data contd.

Let us compare exponential growth rates of few countries?



sns.lineplot functions with plt.subplots

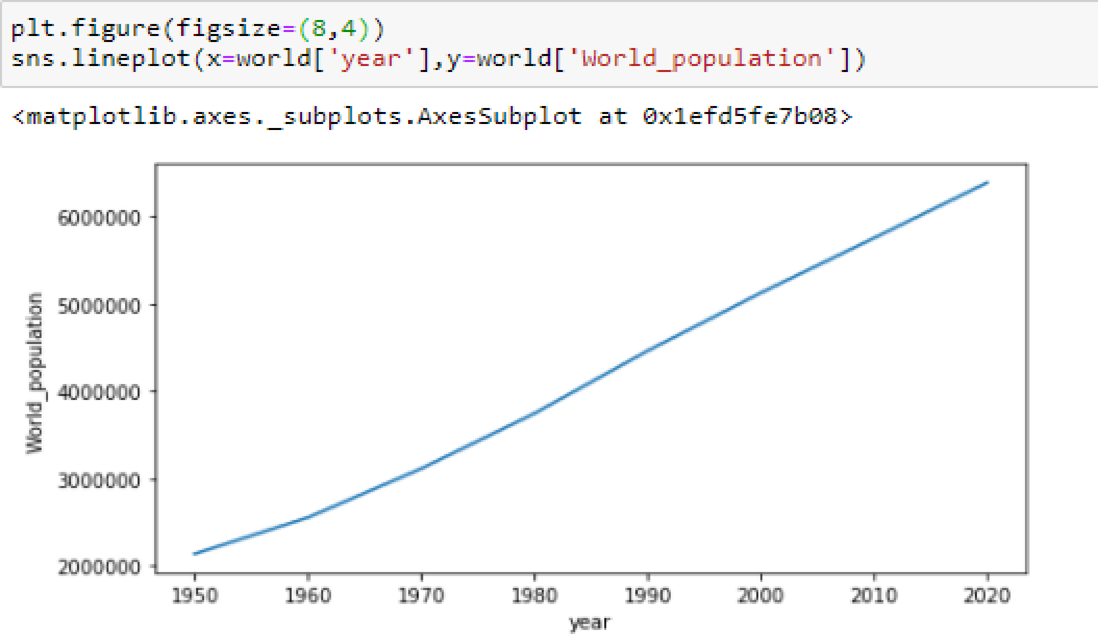
Population growth rate is decreasing in above mentioned nations !

It is an interesting question to ponder - What happens in Afghanistan during 1980?
Reference : https://en.wikipedia.org/wiki/Economy_of_Afghanistan

Germany offset by immigration!

Reference:https://en.wikipedia.org/wiki/Population_decline

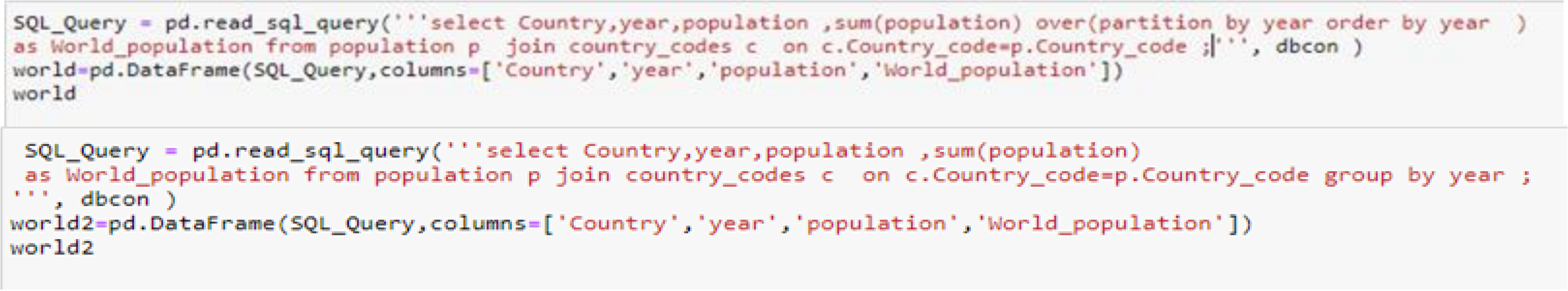
Graphical Inferences from population data contd.



```
plt.figure(figsize=(8,4))
sns.lineplot(x=world['year'],y=world['World_population'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1efd5fe7b08>
```

Analytic/Aggregate function: Sum



```
SQL_Query = pd.read_sql_query('''select Country,year,population ,sum(population) over(partition by year order by year  )
as World_population from population p  join country_codes c  on c.Country_code=p.Country_code ;''', dbcon )
world=pd.DataFrame(SQL_Query,columns=['Country','year','population','World_population'])
world
```

```
 SQL_Query = pd.read_sql_query('''select Country,year,population ,sum(population)
 as World_population from population p join country_codes c  on c.Country_code=p.Country_code group by year ;
''', dbcon )
world2=pd.DataFrame(SQL_Query,columns=['Country','year','population','World_population'])
world2
```

Thank You