RAPPORT PROJET DATA WAREHOUSE

Entrepôt de Données et Reporting

Analysis of the impact on the child mortality of the global population, by exploring five analysis axis : poverty, hunger, health coverage, suicide, geography and population

Realised by : BAALOUACH Sabah N° 08
MIFDAL Ouissal N°39
Option : GDV

academic year: 2022-2023

Table of contents

INTRODUCTION

Since the beginning of the 21st century, one of the main health priorities of the international community is the decline in the mortality of children under the age of 5 . This will be expressed in 2000 in the Millennium Development Goals (MDGs) and, since 2015, in the Sustainable Development Goals (SDGs). One of the eight MDGs was the two-thirds reduction in the global under-five mortality rate between 1990 and 2015.

Although this target has not been met, efforts have been made to halve the infant mortality ratio between 1990 and 2015 (United Nations [UN], 2016). Today, the risk of dying before the 5th birthday is 38 per 1,000 live births (UN, 2019c). This particular focus on mortality of children under the age of 5 is also reflected in the SDGs. One of the SDG targets is to achieve, in all countries, a child-to-child mortality of 25 per 1,000 live births before 2030 (UN, 2018a). Given these numerical targets, set by the United Nations, infant-child mortality has always been a highly monitored health indicator.

The goal of this study is to better understand the link between infant mortality, suicide, hunger, poverty, and health coverage, as well as their evolution over the years.

Data Sources

World Population Dataset : T*his Dataset present the historical population data for every Country/Territory in the world by different parameters like Area Size of the Country/Territory, Name of the Continent, Name of the Capital, Density, Population Growth Rate, Ranking based on Population, World Population Percentage, etc.*

https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset

Global Child Mortality Rate: *This dataset contains data of 197 countries from 1967 to 2020.*

https://www.kaggle.com/datasets/drateendrajha/global-child-mortality-rate

Global Poverty and Inequality Data: *Global Data from Luxembourg Income Study Covering 50+ years and countries.*

https://www.kaggle.com/datasets/stetsondone/global-poverty-and-inequality-data

World Health Statistics 2020|Complete|Geo-Analysis: *The dataset was filtered to increase user readability and create amazing and beautiful visualizations and EDA's.*

https://www.kaggle.com/datasets/utkarshxy/who-worldhealth-statistics-2020-complete

Tools List

Python

Python is the open-source programming language most used by computer scientists. This has propelled itself to the top of infrastructure management, data analysis and software development.

Essential Libraries and Tools:

Pandas

Pandas is a Python library for data processing and analysis. It is built around a data structure called DataFrame which is modeled on the R DataFrame. Simply put, a DataFrame pandas is a table similar to an Excel spreadsheet. Pandas provides a wide range of methods to modify and operate on this table; in particular, it accepts SQL queries and table joins. Another valuable tool provided by pandas is its ability to ingest from a wide variety of file formats and databases, such as SQL, Excel files and Comma-Separated Values (CSV) files.

NumPy

NumPy is one of the fundamental packages for scientific calculation in Python. It contains features for multidimensional arrays, a high-level mathematical function such as linear algebra operations and Fourier transform, and pseudo-random number generators.

Matplotlib

Matplotlib is the library that allows to visualize our Datasets, our functions, our results in the form of graphs, curves, and point clouds.

Scipy

Scipy (Scientific Python) is an open-source library that helps in the computation of complex mathematical or scientific problems. It has a built-in mathematical function and libraries that can be used in science and engineering to resolve different kinds of problems.

PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads

Talend

Talend is an ETL (Extract Transform and Load) software for extracting, transforming, and loading data. Open source, this Java-based tool is widely used in business for managing data flows. Talend also has a part ESB (Company Service Bus). Talend is an ETL that allows you to extract data from a source, modify that data, and then reload it to a destination. The source and destination of the data can be a database, a web service, a csv file. and many others... Talend can therefore be used in any context where data is conveyed.

Microsoft Power BI

Microsoft Power BI is a business intelligence platform that provides users with tools to aggregate, analyze, visualize, and share data. It is used to transform raw data from an enterprise into information used in decision-making. It can help link disparate datasets, transform, and clean data into a data model, and create tables or graphs to provide visual representations of data.

All of this can be shared with other Power BI users within the organization. Power BI can also provide dashboards for administrators or 6 managers, allowing management to get a better idea of the status of services.

Key Performances Indicators [KPI]

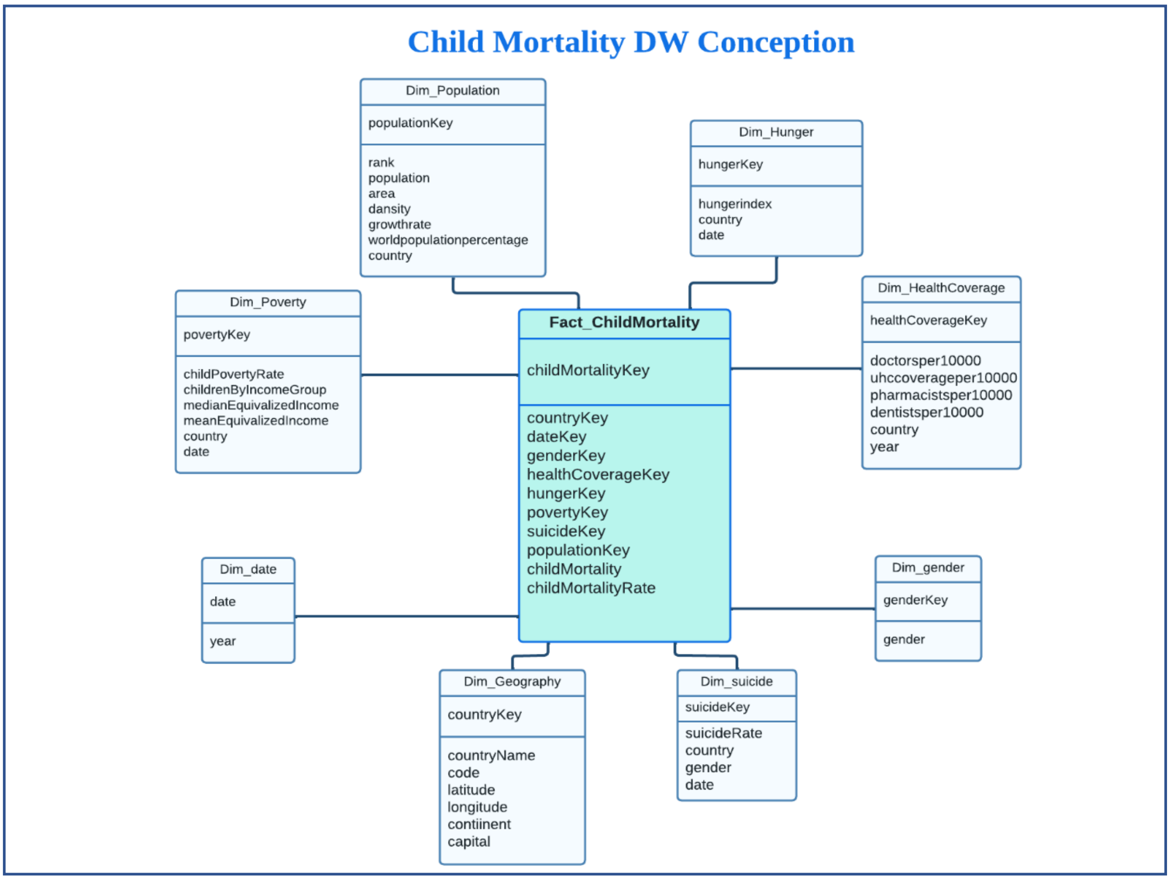
Indicator index	Indicator	Description
I1	Which country has the highest/lowest child mortality rate?	Identify the countries with the highest/lowest child mortality rate .
I2	Which years had the highest/lowest child suicide rates?	Identify the year with the highest/lowest child suicide rates .
I3	What is the impact of poverty on the distribution of child mortality by latitude and longitude?	identify the distribution of infant mortality according to the poverty criterion in the map
I4	what is the sum of density by country?	calculate the sum of population density by country

Dimensional Matrix:

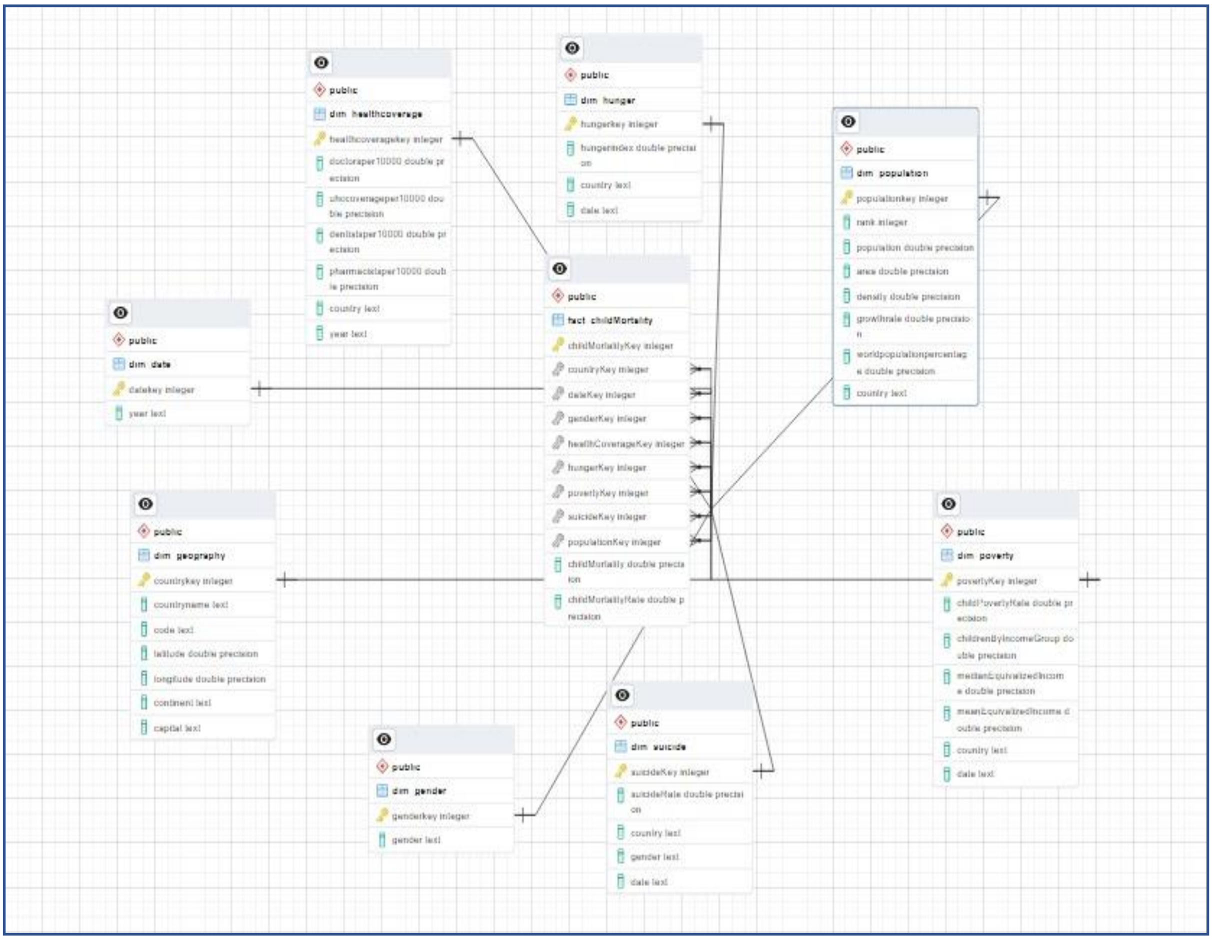
INDICATORS / Axis of analysis	I1	I2	I3	I4
Country	X	X	X	X
Date (year)		X		
Population			X	
Gender	X			X
Poverty			X	
ChildMortalityRate	X			
ChildPovertyRate				
Latitude			X	
Longitude			X	
SuicideRate		X		
Densite				X

Data Warehouse Design

In this part, we will be designing and implementing the data warehouse using the star schema, which is the most simplistic and widely used approach to develop data warehouses. The data warehouse will contain one fact table that is Fact_ChildMortality and eight dimensions: Date, Gender, Geography, Population , Hunger, Suicide, Poverty and Health Coverage . The diagram below represents the approached schema:



In this part of the project, we will implement the data warehouse on PostgreSQL under the name childMortalityDW, we will use the star model, the attributes of each table are represented in the diagram below:



Data Collection and Processing:

Collecting data is nowadays so important to deliver a good, relevant, and consistent analysis. The more important the data, the more difficult it becomes to collect. Indeed, there are several reasons that make data collection difficult, and in our field of study, little data is collected and extracted every day, hence the need to look for other alternatives to collect relevant information.

First of all, it was necessary to go through the data cleansing process to deal with the missing values that exist at the level of each data set, and to be able to analyze the relationship, if any, between infant mortality and factors (poverty, health coverage, hunger and suicide), we performed a data extraction using a python script to obtain the data we will need during the analysis on the infant mortality axis.

After cleaning the data, the following question arises: What would be the link that would make the join between all the tables on which I will do my study? On the one hand, it is possible to have the join based on the identifier of each country, but this will be effective only when the purpose of the analysis concerns more the geographical dimension. On the other hand, the join can be the identifier of the "Date" but, also, this will be a good solution only if the project is more oriented to the date dimension. Thus , the join can be the identifier of the «Gender» but, also, this will be a good solution only if the project is more oriented towards the gender dimension. But because the data sources allow me to get a sense of the three dimensions at once, then I have the opportunity to do an analysis that covers all three tracks.

As a result, it was necessary to create a column that will make it possible to create a link between the different databases and in order to ensure the uniqueness of the column, it will be the result of a concatenation between the key identifying the date and that of each country.

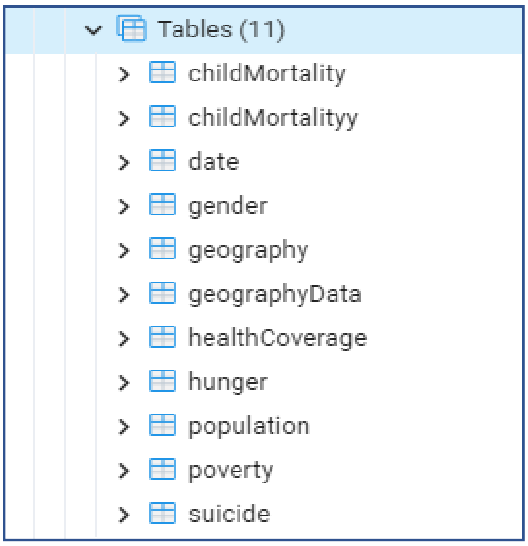
See the associated notebook in appendix

ETL process

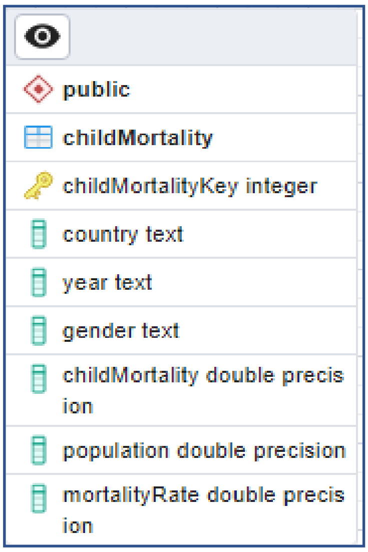
Database Creation

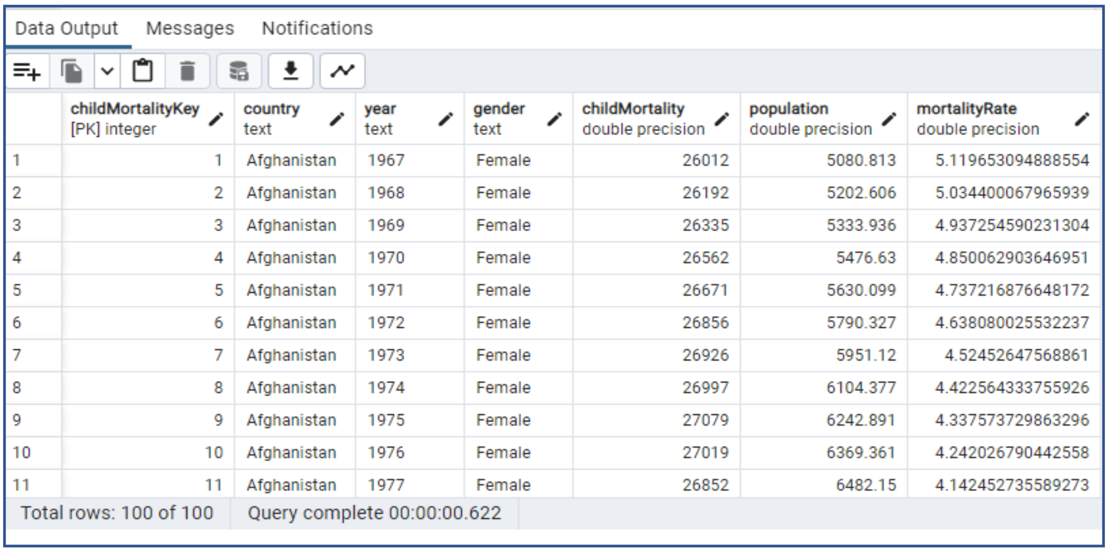
This phase consists of the creation of a **"childMortalityDB"** source database which contains in the form of tables all our files concerning infant mortality and the creation of a **"childMortalityDW"** destination database which will contain, the dimension tables and the de facto table while specifying the fields and their types.

Database: childMortalityDB



- *Table: childMortality*

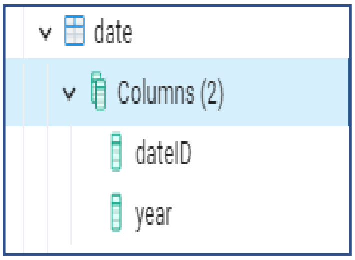


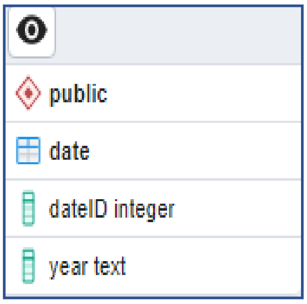


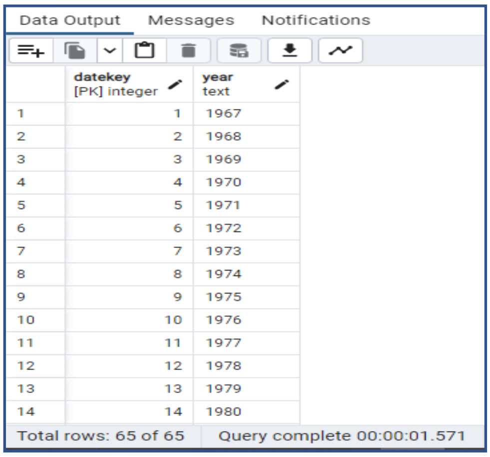
	childMortalityKey [PK] integer	country text	year text	gender text	childMortality double precision	population double precision	mortalityRate double precision
1	1	Afghanistan	1967	Female	26012	5080.813	5.119653094888554
2	2	Afghanistan	1968	Female	26192	5202.606	5.034400067965939
3	3	Afghanistan	1969	Female	26335	5333.936	4.937254590231304
4	4	Afghanistan	1970	Female	26562	5476.63	4.850062903646951
5	5	Afghanistan	1971	Female	26671	5630.099	4.737216876648172
6	6	Afghanistan	1972	Female	26856	5790.327	4.638080025532237
7	7	Afghanistan	1973	Female	26926	5951.12	4.52452647568861
8	8	Afghanistan	1974	Female	26997	6104.377	4.422564333755926
9	9	Afghanistan	1975	Female	27079	6242.891	4.337573729863296
10	10	Afghanistan	1976	Female	27019	6369.361	4.242026790442558
11	11	Afghanistan	1977	Female	26852	6482.15	4.142452735589273

Total rows: 100 of 100 Query complete 00:00:00.622

- *Table: Date*



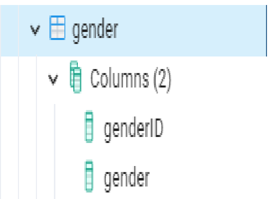


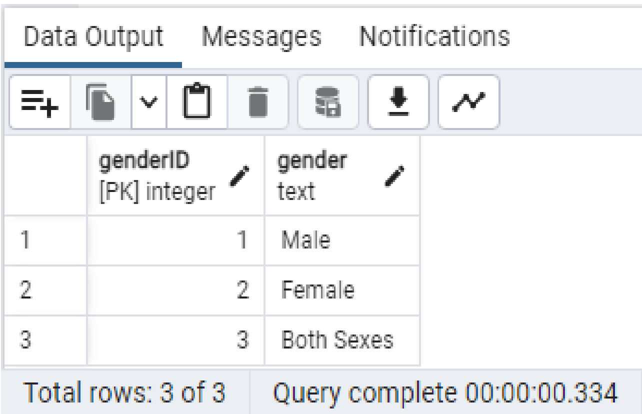


	datekey [PK] integer	year text
1	1	1967
2	2	1968
3	3	1969
4	4	1970
5	5	1971
6	6	1972
7	7	1973
8	8	1974
9	9	1975
10	10	1976
11	11	1977
12	12	1978
13	13	1979
14	14	1980

Total rows: 65 of 65 Query complete 00:00:01.571

- *Table: Gender*

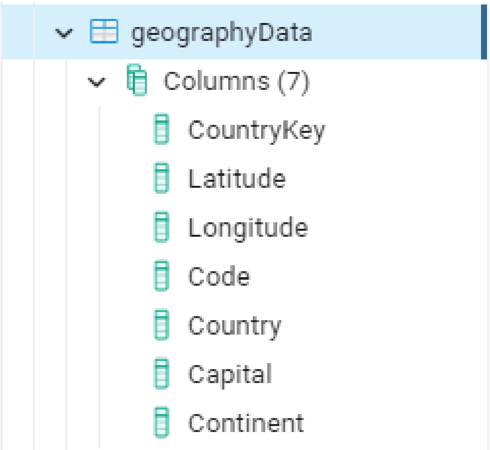


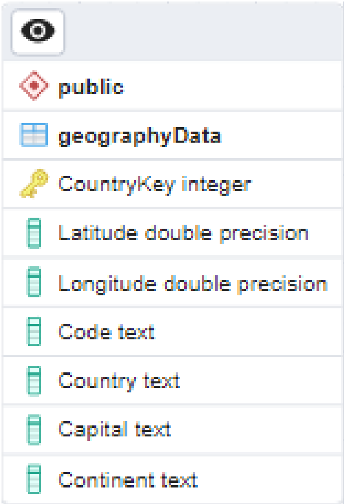


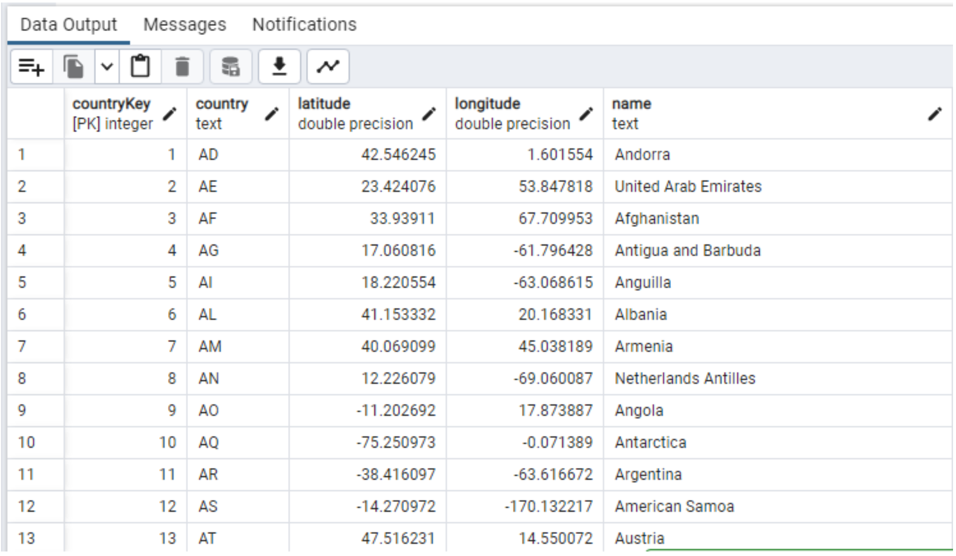
	genderID [PK] integer	gender text
1	1	Male
2	2	Female
3	3	Both Sexes

Total rows: 3 of 3 Query complete 00:00:00.334

- *Table: GeographyData*

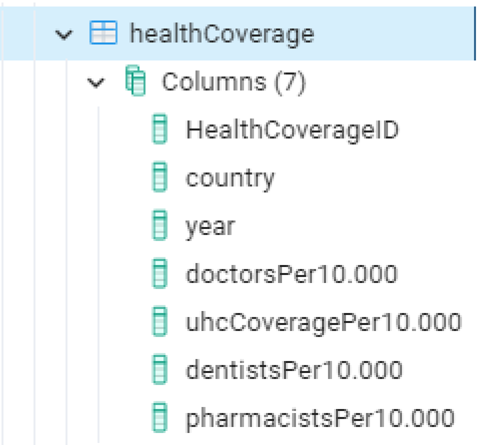


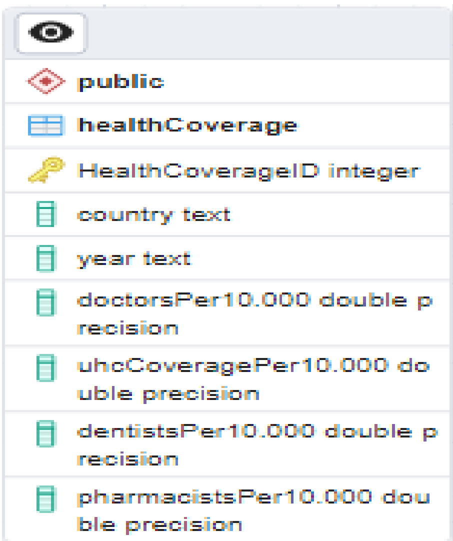



Data Output Messages Notifications

	countryKey [PK] integer	country text	latitude double precision	longitude double precision	name text
1	1	AD	42.546245	1.601554	Andorra
2	2	AE	23.424076	53.847818	United Arab Emirates
3	3	AF	33.93911	67.709953	Afghanistan
4	4	AG	17.060816	-61.796428	Antigua and Barbuda
5	5	AI	18.220554	-63.068615	Anguilla
6	6	AL	41.153332	20.168331	Albania
7	7	AM	40.069099	45.038189	Armenia
8	8	AN	12.226079	-69.060087	Netherlands Antilles
9	9	AO	-11.202692	17.873887	Angola
10	10	AQ	-75.250973	-0.071389	Antarctica
11	11	AR	-38.416097	-63.616672	Argentina
12	12	AS	-14.270972	-170.132217	American Samoa
13	13	AT	47.516231	14.550072	Austria

- *Table: HealthCoverage*



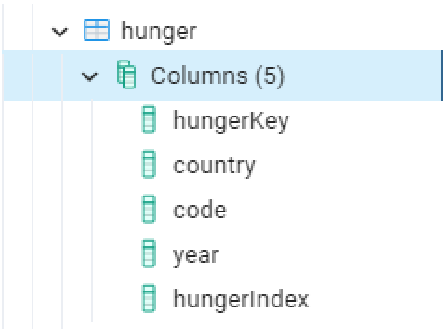


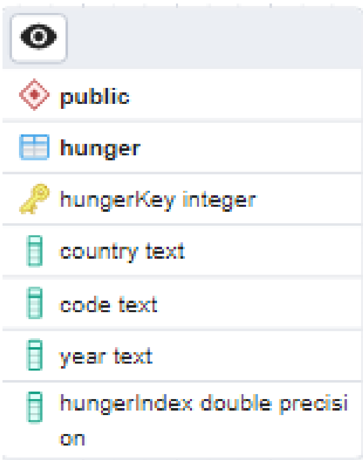
Data Output | Messages | Notifications

	HealthCoverageID [PK] integer	country text	year text	doctorsPer10.000 double precision	uhcCoveragePer10.000 double precision	dentistsPer10.000 double precision	pharmacistsPer10.000 double precision
1	1	Afghanistan	2016	2.78	67.08910891089108	0.034	0.4
2	2	Afghanistan	2015	2.85	34	0.036	0.
3	3	Afghanistan	2014	2.98	67.08910891089108	0.033	0.5
4	4	Afghanistan	2013	2.85	67.08910891089108	4.101533773489081	4.26790749185667
5	5	Afghanistan	2012	2.41	67.08910891089108	4.101533773489081	4.26790749185667
6	6	Afghanistan	2011	2.52	67.08910891089108	4.101533773489081	4.26790749185667
7	7	Afghanistan	2010	2.37	67.08910891089108	4.101533773489081	4.26790749185667
8	8	Afghanistan	2009	2.13	67.08910891089108	4.101533773489081	4.26790749185667
9	9	Afghanistan	2008	1.74	67.08910891089108	0.14	0.
10	10	Afghanistan	2007	1.74	67.08910891089108	0.12	0.3
11	11	Afghanistan	2006	1.6	67.08910891089108	0.13	0.3
12	12	Afghanistan	2001	1.9	67.08910891089108	4.101533773489081	4.26790749185667
13	13	Albania	2016	12.16	67.08910891089108	4.101533773489081	4.26790749185667
14	14	Albania	2013	12.77	67.08910891089108	4.101533773489081	4.26790749185667
15	15	Albania	2012	12.68	67.08910891089108	4.101533773489081	4.26790749185667
16	16	Albania	2011	12.22	67.08910891089108	4.101533773489081	4.26790749185667
17	17	Albania	2010	12.35	67.08910891089108	4.101533773489081	4.26790749185667

Total rows: 100 of 100 | Query complete 00:00:00.360 | Ln 1,

- *Table: Hunger*





Data Output Messages Notifications

	hungerKey [PK] integer	country text	code text	year text	hungerIndex double precision
1	1	Afghanistan	AFG	2000	50.9
2	2	Afghanistan	AFG	2006	42.7
3	3	Afghanistan	AFG	2012	34.3
4	4	Afghanistan	AFG	2021	28.3
5	5	Albania	ALB	2000	20.7
6	6	Albania	ALB	2006	15.9
7	7	Albania	ALB	2012	8.8
8	8	Albania	ALB	2021	6.2
9	9	Algeria	DZA	2000	14.5
10	10	Algeria	DZA	2006	11.7
11	11	Algeria	DZA	2012	8.9
12	12	Algeria	DZA	2021	6.9
13	13	Angola	AGO	2000	65
14	14	Angola	AGO	2006	46.9
15	15	Angola	AGO	2012	27.8
16	16	Angola	AGO	2021	26
17	17	Argentina	ARG	2000	6.4

Total rows: 100 of 100 Query complete 00:00:00.397

- *Table :Population*

population
- Columns (11)
  - populationKey
  - rank
  - code
  - country
  - capital
  - continent
  - population
  - area
  - density
  - growthRate
  - worldPopulationPercentageou

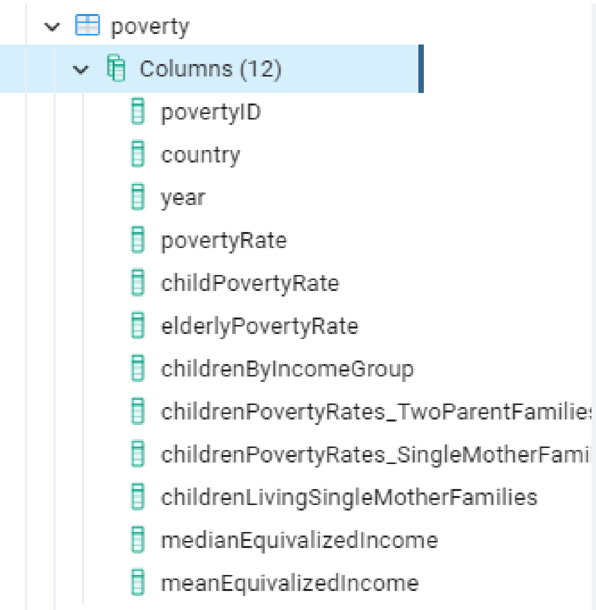
public
population
- populationKey integer
- rank integer
- code text
- country text
- capital text
- continent text
- population bigint
- area double precision
- density double precision
- growthRate double precision
- worldPopulationPercentageou double precision

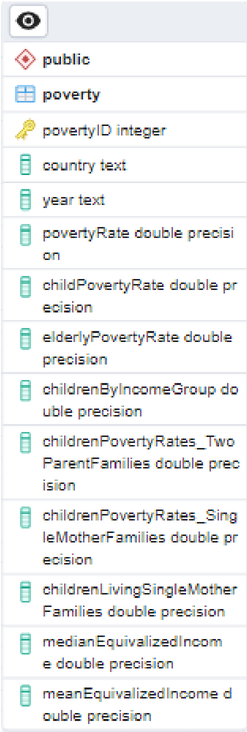
Data Output Messages Notifications

	CountryKey [PK] integer	Latitude double precision	Longitude double precision	Code text	Country text	Capital text	Continent text
1	1	42.546245	1.601554	AND	Andorra	Andorra la Vella	Europe
2	2	23.424076	53.847818	ARE	United Arab Emirates	Abu Dhabi	Asia
3	3	33.93911	67.709953	AFG	Afghanistan	Kabul	Asia
4	4	17.060816	-61.796428	ATG	Antigua and Barbuda	Saint John's	North America
5	5	18.220554	-63.068615	AIA	Anguilla	The Valley	North America
6	6	41.153332	20.168331	ALB	Albania	Tirana	Europe
7	7	40.069099	45.038189	ARM	Armenia	Yerevan	Asia
8	8	-11.202692	17.873887	AGO	Angola	Luanda	Africa
9	9	-38.416097	-63.616672	ARG	Argentina	Buenos Aires	South America
10	10	-14.270972	-170.132217	ASM	American Samoa	Pago Pago	Oceania
11	11	47.516231	14.550072	AUT	Austria	Vienna	Europe
12	12	-25.274398	133.775136	AUS	Australia	Canberra	Oceania
13	13	12.52111	-69.968338	ABW	Aruba	Oranjestad	North America

Total rows: 100 of 100 Query complete 00:00:01.180

- *Table : Poverty*

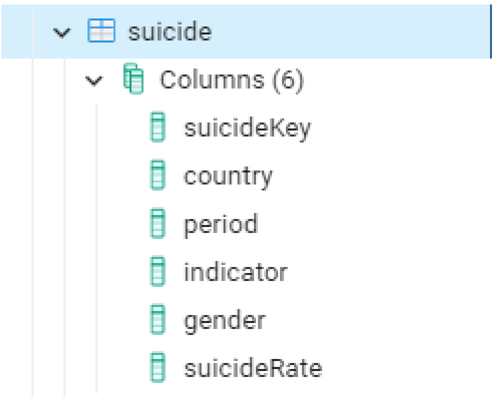


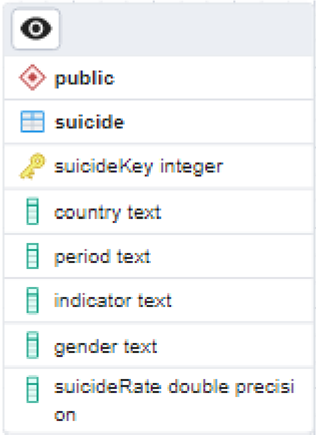


Data Output Messages Notifications

	povertyID [PK] integer	country text	year text	povertyRate double precision	childPovertyRate double precision	elderlyPovertyRate double precision	childrenByIncomeGroup double precision
1	1	Australia	2018	12.574	13.049	25.607	16.176
2	2	Australia	2016	12.441	12.406	25.915	17.755
3	3	Australia	2014	12.465	11.423	26.495	17.2
4	4	Australia	2010	14.059	14.42	33.628	15.494
5	5	Australia	2008	14.235	12.845	37.151	14.832
6	6	Australia	2004	13.302	12.109	28.863	14.515
7	7	Australia	2003	12.504	13.887	22.531	13.799
8	8	Australia	2001	13.175	14.865	22.672	14.26
9	9	Australia	1995	11.57	13.196	21.799	13.216
10	10	Australia	1989	12.317	15.056	24.348	12.126
11	11	Australia	1985	11.914	13.953	24.647	10.666
12	12	Australia	1981	11.408	13.887	13.533	16.207
13	13	Austria	2019	9.981	12.902	10.467	11.163
14	14	Austria	2018	9.011	8.948	9.936	9.912
15	15	Austria	2017	9.302	11.434	9.834	11.518
16	16	Austria	2016	9.637	11.693	8.664	10.94

- *Table : Suicide*

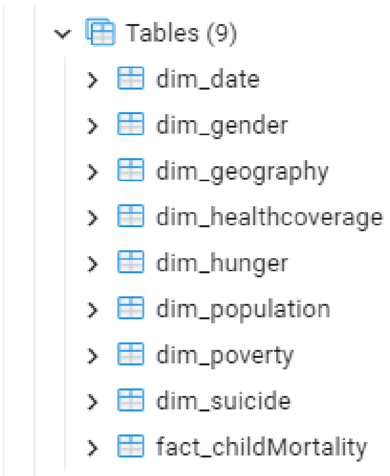




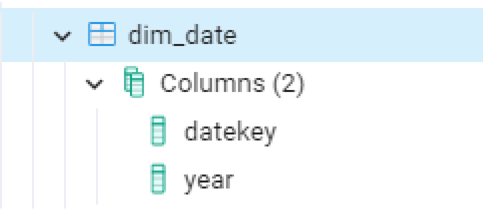
Data Output Messages Notifications

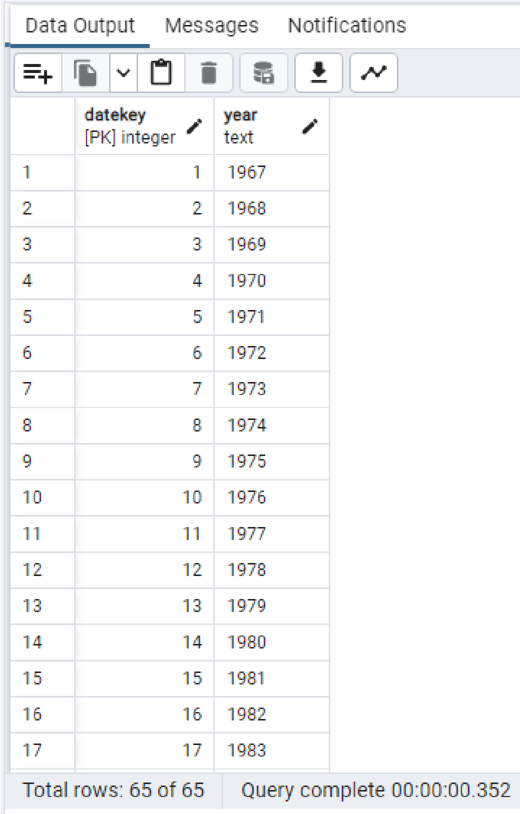
	suicideKey [PK] integer	country text	period text	indicator text	gender text	suicideRate double precision
1	1	Afghanistan	2016	Crude suicide rates (per 100 000 population)	Both sexes	0
2	2	Afghanistan	2016	Crude suicide rates (per 100 000 population)	Male	0
3	3	Afghanistan	2016	Crude suicide rates (per 100 000 population)	Female	0
4	4	Afghanistan	2015	Crude suicide rates (per 100 000 population)	Both sexes	4.8
5	5	Afghanistan	2015	Crude suicide rates (per 100 000 population)	Male	7.8
6	6	Afghanistan	2015	Crude suicide rates (per 100 000 population)	Female	1.5
7	7	Afghanistan	2010	Crude suicide rates (per 100 000 population)	Both sexes	5.1
8	8	Afghanistan	2010	Crude suicide rates (per 100 000 population)	Male	8.6
9	9	Afghanistan	2010	Crude suicide rates (per 100 000 population)	Female	1.4
10	10	Afghanistan	2005	Crude suicide rates (per 100 000 population)	Both sexes	6.3
11	11	Afghanistan	2005	Crude suicide rates (per 100 000 population)	Male	10.8
12	12	Afghanistan	2005	Crude suicide rates (per 100 000 population)	Female	1.5
13	13	Afghanistan	2000	Crude suicide rates (per 100 000 population)	Both sexes	5.7
14	14	Afghanistan	2000	Crude suicide rates (per 100 000 population)	Male	10
15	15	Afghanistan	2000	Crude suicide rates (per 100 000 population)	Female	1
16	16	Albania	2016	Crude suicide rates (per 100 000 population)	Both sexes	0
17	17	Albania	2016	Crude suicide rates (per 100 000 population)	Male	0

Datawarehouse childMortalityDW



- *Table: Date*

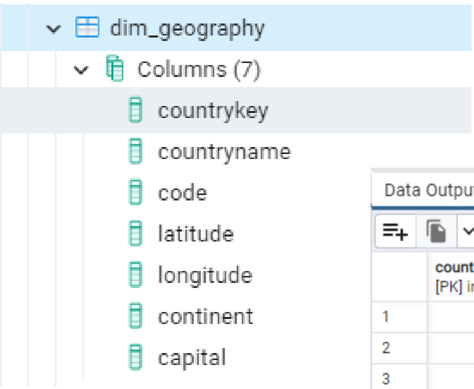


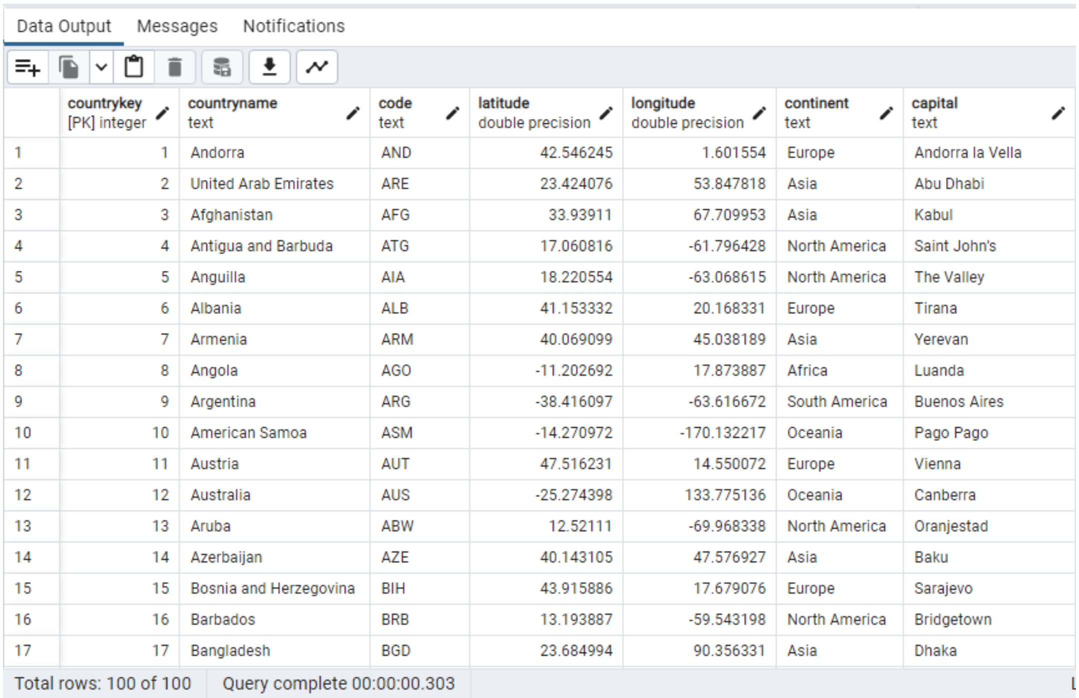


	datekey [PK] integer	year text
1	1	1967
2	2	1968
3	3	1969
4	4	1970
5	5	1971
6	6	1972
7	7	1973
8	8	1974
9	9	1975
10	10	1976
11	11	1977
12	12	1978
13	13	1979
14	14	1980
15	15	1981
16	16	1982
17	17	1983

Total rows: 65 of 65 Query complete 00:00:00.352

- *Table: Geography*





	countrykey [PK] integer	countryname text	code text	latitude double precision	longitude double precision	continent text	capital text
1	1	Andorra	AND	42.546245	1.601554	Europe	Andorra la Vella
2	2	United Arab Emirates	ARE	23.424076	53.847818	Asia	Abu Dhabi
3	3	Afghanistan	AFG	33.93911	67.709953	Asia	Kabul
4	4	Antigua and Barbuda	ATG	17.060816	-61.796428	North America	Saint John's
5	5	Anguilla	AIA	18.220554	-63.068615	North America	The Valley
6	6	Albania	ALB	41.153332	20.168331	Europe	Tirana
7	7	Armenia	ARM	40.069099	45.038189	Asia	Yerevan
8	8	Angola	AGO	-11.202692	17.873887	Africa	Luanda
9	9	Argentina	ARG	-38.416097	-63.616672	South America	Buenos Aires
10	10	American Samoa	ASM	-14.270972	-170.132217	Oceania	Pago Pago
11	11	Austria	AUT	47.516231	14.550072	Europe	Vienna
12	12	Australia	AUS	-25.274398	133.775136	Oceania	Canberra
13	13	Aruba	ABW	12.52111	-69.968338	North America	Oranjestad
14	14	Azerbaijan	AZE	40.143105	47.576927	Asia	Baku
15	15	Bosnia and Herzegovina	BIH	43.915886	17.679076	Europe	Sarajevo
16	16	Barbados	BRB	13.193887	-59.543198	North America	Bridgetown
17	17	Bangladesh	BGD	23.684994	90.356331	Asia	Dhaka

Total rows: 100 of 100 Query complete 00:00:00.303

- *Table: HealthCoverage*

dim_healthcoverage
- Columns (7)
  - healthcoveragekey
  - doctorsper10000
  - uhccoverageper10000
  - dentistsper10000
  - pharmacistsper10000
  - country
  - year

Data Output | Messages | Notifications

	healthcoveragekey [PK] integer	doctorsper10000 double precision	uhccoverageper10000 double precision	dentistsper10000 double precision	pharmacistsper10000 double precision	country text	year text
1	1	2.78	67.08910891089108	0.034	0.47	Afghanistan	2016
2	2	2.85	34	0.036	0.5	Afghanistan	2015
3	3	2.98	67.08910891089108	0.033	0.51	Afghanistan	2014
4	4	2.85	67.08910891089108	4.101533773489081	4.267907491856677	Afghanistan	2013
5	5	2.41	67.08910891089108	4.101533773489081	4.267907491856677	Afghanistan	2012
6	6	2.52	67.08910891089108	4.101533773489081	4.267907491856677	Afghanistan	2011
7	7	2.37	67.08910891089108	4.101533773489081	4.267907491856677	Afghanistan	2010
8	8	2.13	67.08910891089108	4.101533773489081	4.267907491856677	Afghanistan	2009
9	9	1.74	67.08910891089108	0.14	0.3	Afghanistan	2008
10	10	1.74	67.08910891089108	0.12	0.31	Afghanistan	2007
11	11	1.6	67.08910891089108	0.13	0.31	Afghanistan	2006
12	12	1.9	67.08910891089108	4.101533773489081	4.267907491856677	Afghanistan	2001
13	13	12.16	67.08910891089108	4.101533773489081	4.267907491856677	Albania	2016
14	14	12.77	67.08910891089108	4.101533773489081	4.267907491856677	Albania	2013
15	15	12.68	67.08910891089108	4.101533773489081	4.267907491856677	Albania	2012
16	16	12.22	67.08910891089108	4.101533773489081	4.267907491856677	Albania	2011

Total rows: 100 of 100 | Query complete 00:00:00.253 | Ln 1, Col 1

- *Table: Hunger*

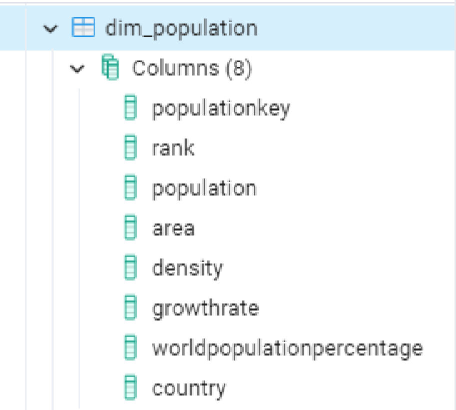
dim_hunger
- Columns (4)
  - hungerkey
  - hungerindex
  - country
  - date

Data Output | Messages | Notifications

	hungerkey [PK] integer	hungerindex double precision	country text	date text
1	1	50.9	Afghanistan	2000
2	2	42.7	Afghanistan	2006
3	3	34.3	Afghanistan	2012
4	4	28.3	Afghanistan	2021
5	5	20.7	Albania	2000
6	6	15.9	Albania	2006
7	7	8.8	Albania	2012
8	8	6.2	Albania	2021
9	9	14.5	Algeria	2000
10	10	11.7	Algeria	2006
11	11	8.9	Algeria	2012
12	12	6.9	Algeria	2021
13	13	65	Angola	2000
14	14	46.9	Angola	2006
15	15	27.8	Angola	2012
16	16	26	Angola	2021
17	17	6.4	Argentina	2000

Total rows: 100 of 100 | Query complete 00:00:00.426

- *Table: Population*

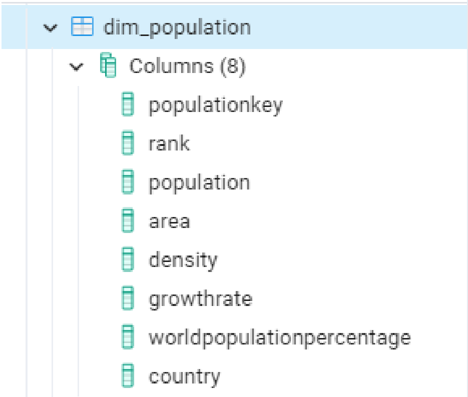


Data Output Messages Notifications

	populationkey [PK] integer	rank integer	population double precision	area double precision	density double precision	growthrate double precision	worldpopulationpercentage double precision	c te
1	1	36	41128771	652230	63.0587	1.0257	0.52	A
2	2	138	2842321	28748	98.8702	0.9957	0.04	A
3	3	34	44903225	2381741	18.8531	1.0164	0.56	A
4	4	213	44273	199	222.4774	0.9831	0	A
5	5	203	79824	468	170.5641	1.01	0	A
6	6	42	35588987	1246700	28.5466	1.0315	0.45	A
7	7	224	15857	91	174.2527	1.0066	0	A
8	8	201	93763	442	212.1335	1.0058	0	A
9	9	33	45510318	2780400	16.3683	1.0052	0.57	A
10	10	140	2780469	29743	93.4831	0.9962	0.03	A
11	11	198	106445	180	591.3611	0.9991	0	A
12	12	55	26177413	7692024	3.4032	1.0099	0.33	A
13	13	99	8939617	83871	106.5877	1.002	0.11	A
14	14	91	10358074	86600	119.6082	1.0044	0.13	A
15	15	176	409984	13943	29.4043	1.0051	0.01	B
16	16	154	1472233	765	1924.4876	1.0061	0.02	B

Total rows: 100 of 100 Query complete 00:00:00.557 Ln 1, Col 1

- *Table: Poverty*

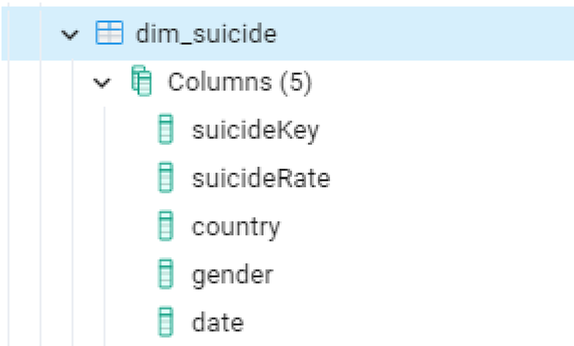


Data Output Messages Notifications

	populationkey [PK] integer	rank integer	population double precision	area double precision	density double precision	growthrate double precision	worldpopulationpercentage double precision	c te
1	1	36	41128771	652230	63.0587	1.0257	0.52	A
2	2	138	2842321	28748	98.8702	0.9957	0.04	A
3	3	34	44903225	2381741	18.8531	1.0164	0.56	A
4	4	213	44273	199	222.4774	0.9831	0	A
5	5	203	79824	468	170.5641	1.01	0	A
6	6	42	35588987	1246700	28.5466	1.0315	0.45	A
7	7	224	15857	91	174.2527	1.0066	0	A
8	8	201	93763	442	212.1335	1.0058	0	A
9	9	33	45510318	2780400	16.3683	1.0052	0.57	A
10	10	140	2780469	29743	93.4831	0.9962	0.03	A
11	11	198	106445	180	591.3611	0.9991	0	A
12	12	55	26177413	7692024	3.4032	1.0099	0.33	A
13	13	99	8939617	83871	106.5877	1.002	0.11	A
14	14	91	10358074	86600	119.6082	1.0044	0.13	A
15	15	176	409984	13943	29.4043	1.0051	0.01	B
16	16	154	1472233	765	1924.4876	1.0061	0.02	B

Total rows: 100 of 100 Query complete 00:00:00.557 Ln 1, Col 1

- *Table: Suicide*



Data Output | Messages | Notifications

	suicideKey [PK] integer	suicideRate double precision	country text	gender text	date text
1	1	0	Afghanistan	Both sexes	2016
2	2	0	Afghanistan	Male	2016
3	3	0	Afghanistan	Female	2016
4	4	4.8	Afghanistan	Both sexes	2015
5	5	7.8	Afghanistan	Male	2015
6	6	1.5	Afghanistan	Female	2015
7	7	5.1	Afghanistan	Both sexes	2010
8	8	8.6	Afghanistan	Male	2010
9	9	1.4	Afghanistan	Female	2010
10	10	6.3	Afghanistan	Both sexes	2005
11	11	10.8	Afghanistan	Male	2005
12	12	1.5	Afghanistan	Female	2005
13	13	5.7	Afghanistan	Both sexes	2000
14	14	10	Afghanistan	Male	2000
15	15	1	Afghanistan	Female	2000
16	16	0	Albania	Both sexes	2016
17	17	0	Albania	Male	2016

Total rows: 100 of 100 | Query complete 00:00:00.374

- *Table: Fact_childMortality*

fact_childMortality
- Columns (11)
  - childMortalityKey
  - countryKey
  - dateKey
  - genderKey
  - healthCoverageKey
  - hungerKey
  - povertyKey
  - suicideKey
  - populationKey
  - childMortality
  - childMortalityRate

Data Output Messages Notifications

	childMortalityKey [PK] integer	countryKey integer	dateKey integer	genderKey integer	healthCoverageKey integer	hungerKey integer	povertyKey integer	suicideKey integer
1	1	29	34	2	327	53	59	343
2	2	29	34	2	327	53	59	344
3	3	29	34	2	327	53	59	345
4	4	29	34	2	327	53	60	343
5	5	29	34	2	327	53	60	344
6	6	29	34	2	327	53	60	345
7	7	29	34	2	327	53	61	343
8	8	29	34	2	327	53	61	344
9	9	29	34	2	327	53	61	345
10	10	29	34	2	327	53	62	343
11	11	29	34	2	327	53	62	344
12	12	29	34	2	327	53	62	345
13	13	29	34	2	327	53	63	343
14	14	29	34	2	327	53	63	344
15	15	29	34	2	327	53	63	345
16	16	29	34	2	327	54	59	343

Total rows: 100 of 100 Query complete 00:00:00.357

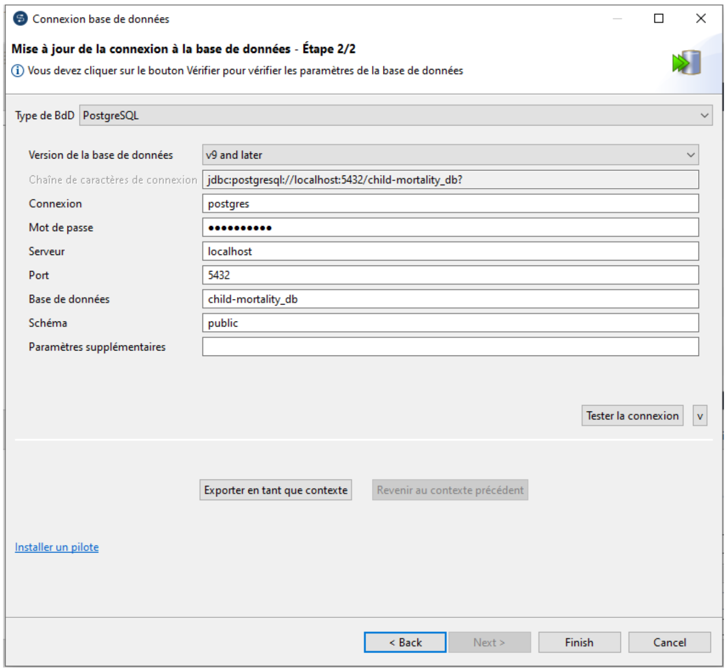
ETL Pipeline:

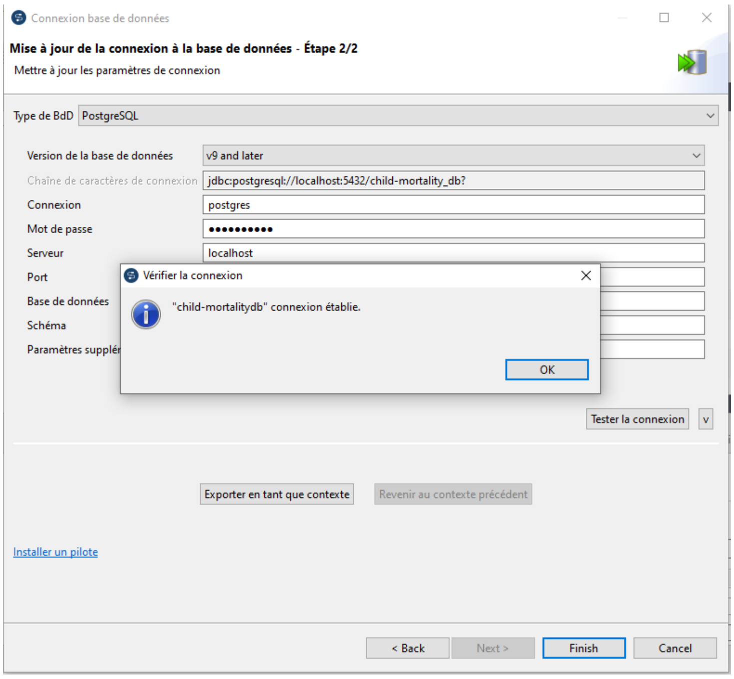
Extract, Transform, Load is the process used to extract data from different sources, transform it and load it on a destination system end user can access. For this project we will be extracting the data from several flat csv sources and load it into the Project Database that was created beforehand on **PostgreSQL** . Next, we will use a data pipeline to migrate our data from the database to the Datawarehouse.

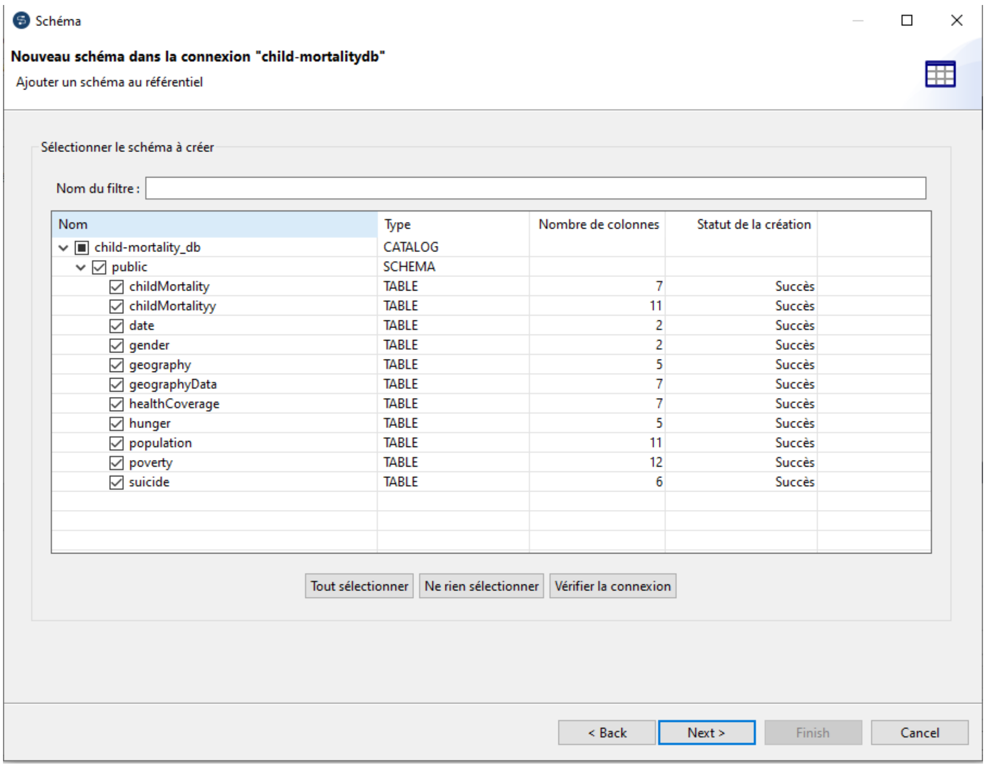
we will be using **Talend** to go through the ETL process , we would have to fill all the dimensions before proceeding to fill the fact table.

Then ,after the creation of the data warehouse, taking care of its supply following the ETL process (Extraction, Transformation, Loading) Above, the results of the ETL process and the result on Talend:

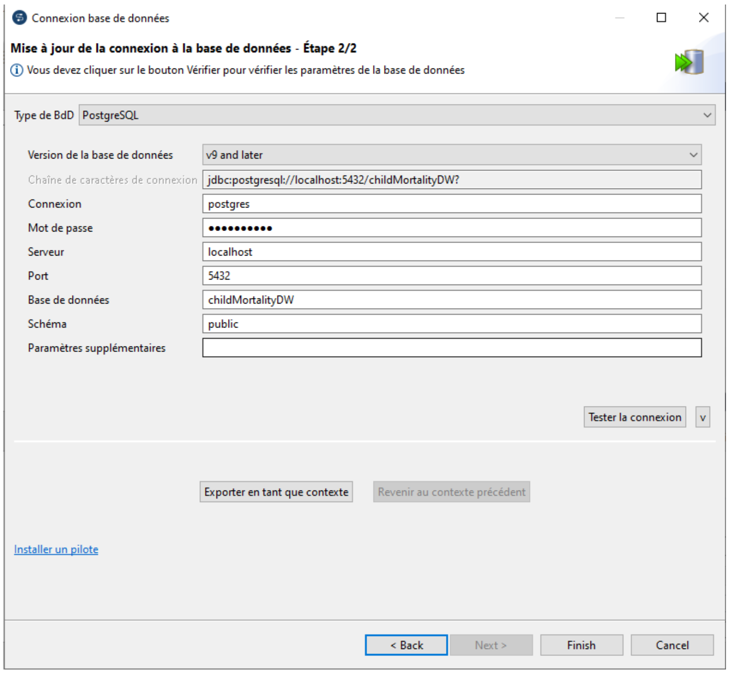
1. First, in Talend ,we connect to the "child-mortalitydb" database:

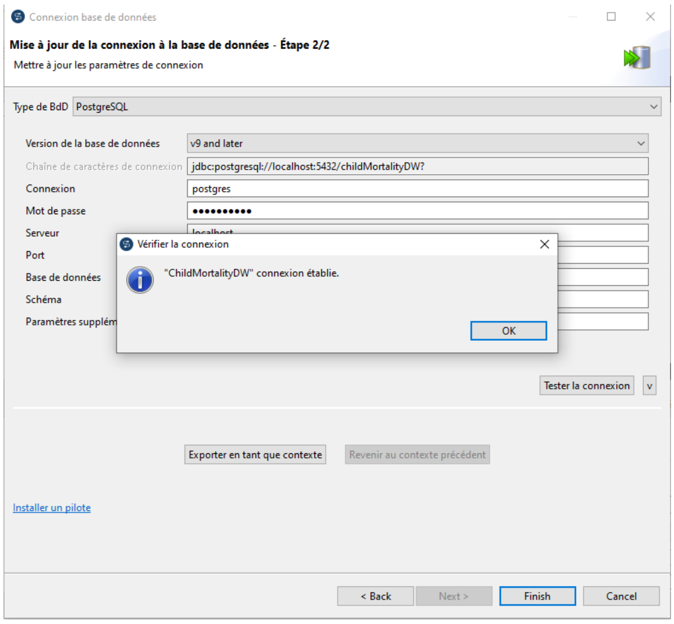


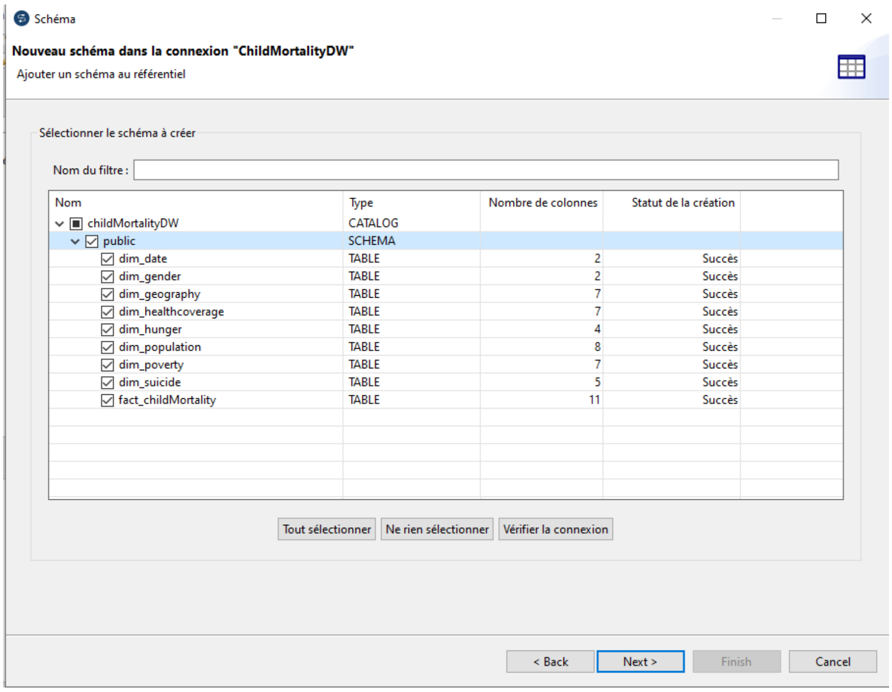




2. Similarly, we do the connection to the Datawarehouse "childMortalityDW"







3. After logging in with database and datawarehouse, we pass to do ETL in Talend using:

-tDBInput :

tDBInput allows you to extract data from a database. This component works with different databases according to our selection. The tDBInput configuration runs in the Jobs Standard framework

-tDBOutput :

tDBOutput allows you to write, update, modify or delete data from a database.

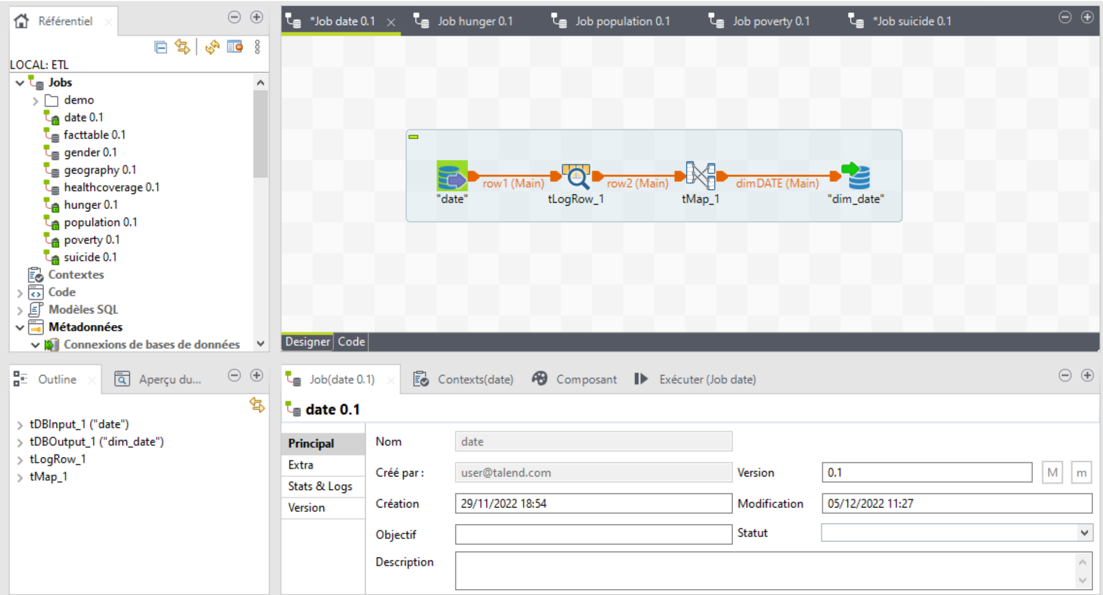
-tMap:

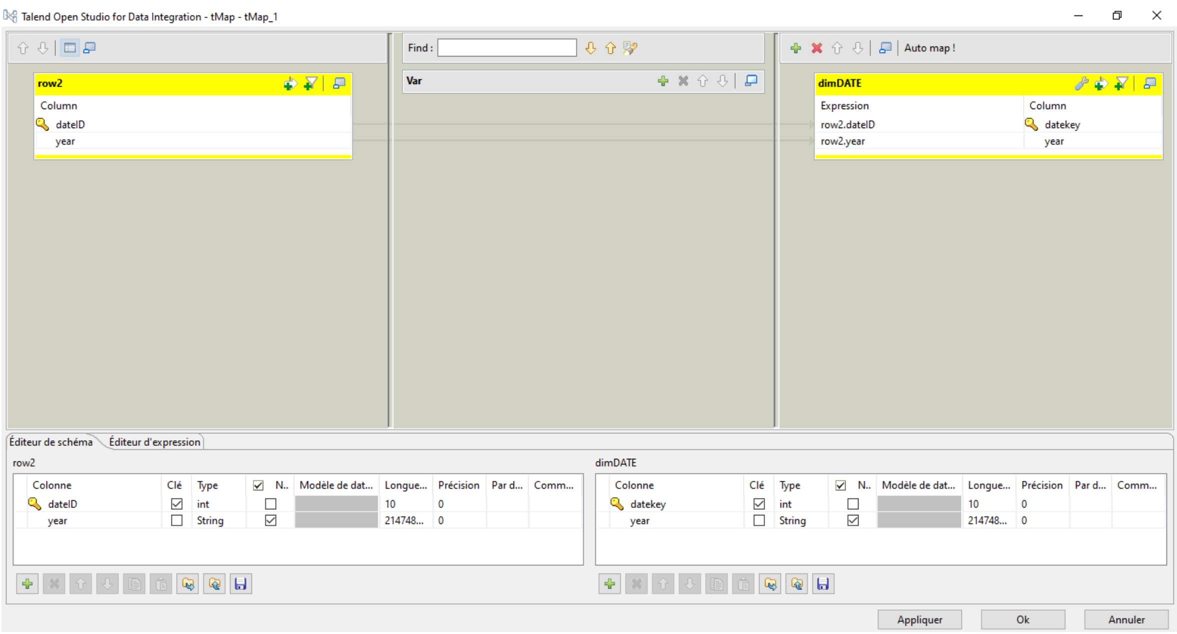
tMap is the most important and powerful component of Talend. It allows to perform multi operations (joins, transformations, filters, rejects...) The expressions used are in Java.

-tLogRow:

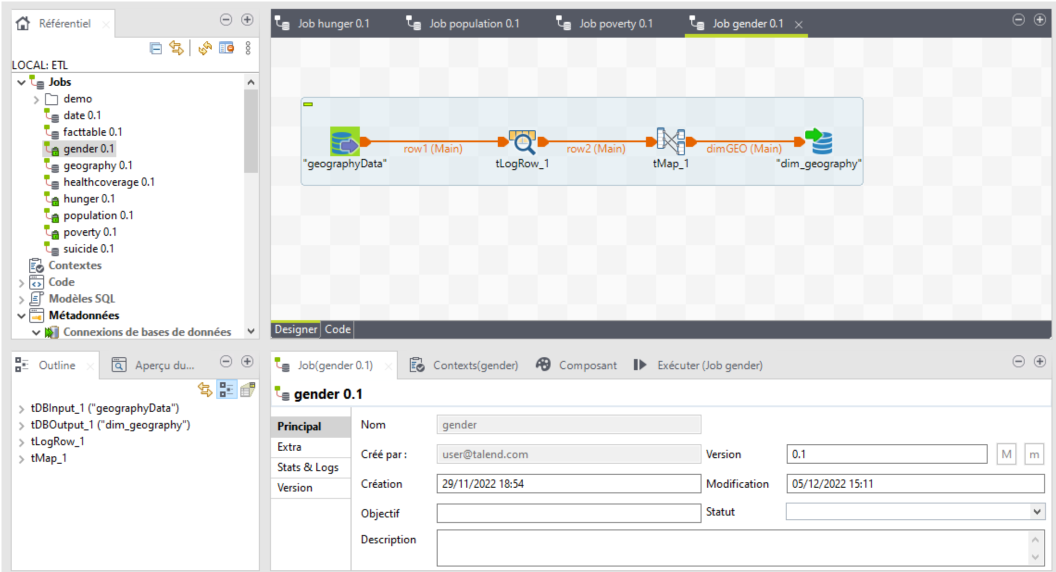
tLogRow is an excellent debugging tool. it allows to display by line and It sends the data to the console.

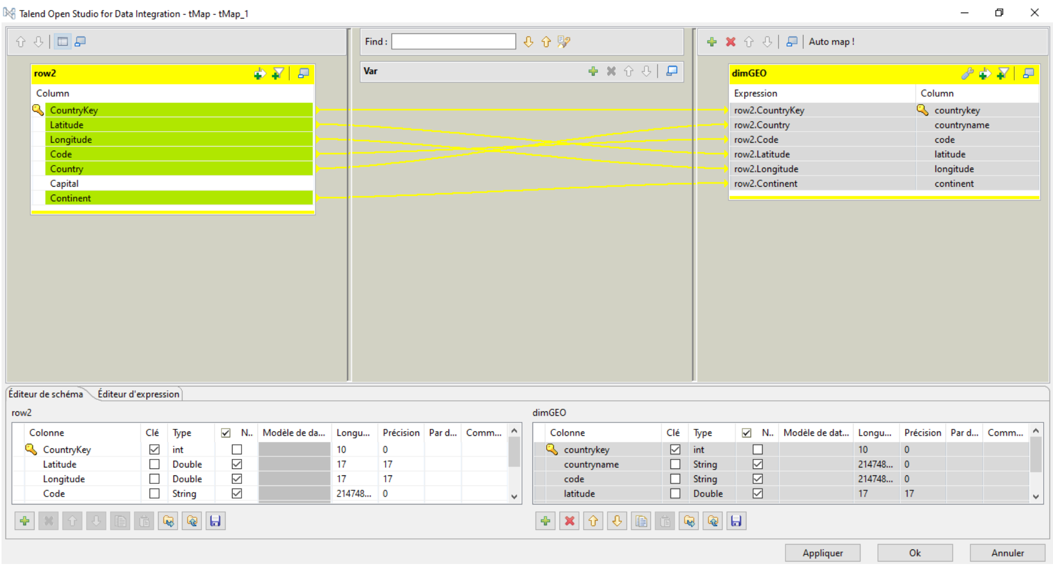
- *Date Dimension*



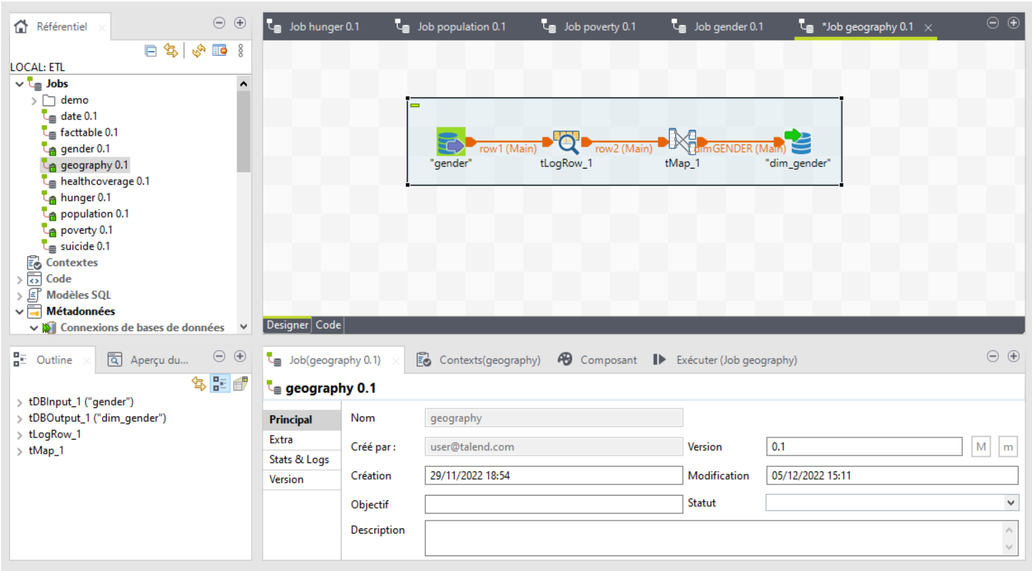


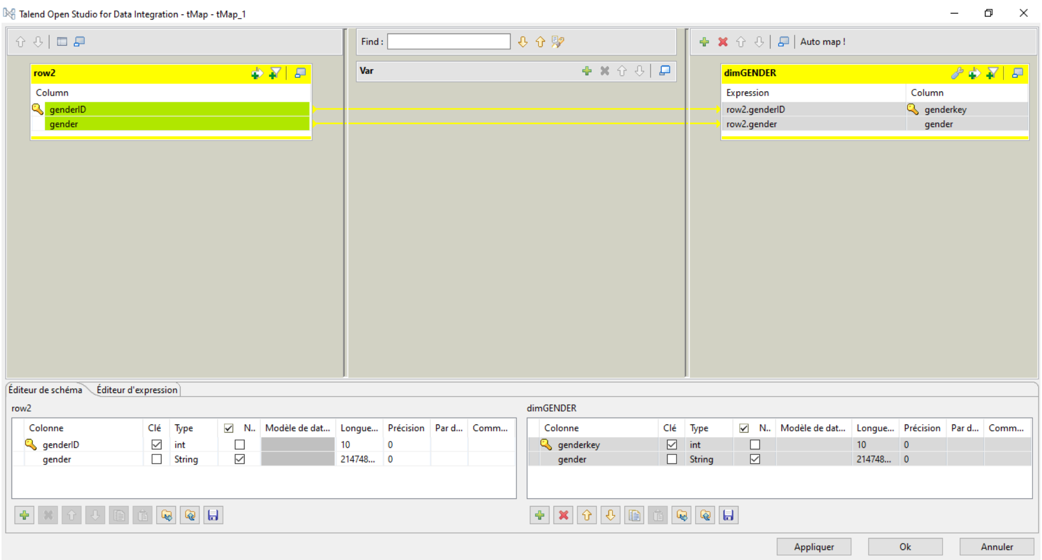
- *Geography Dimension*





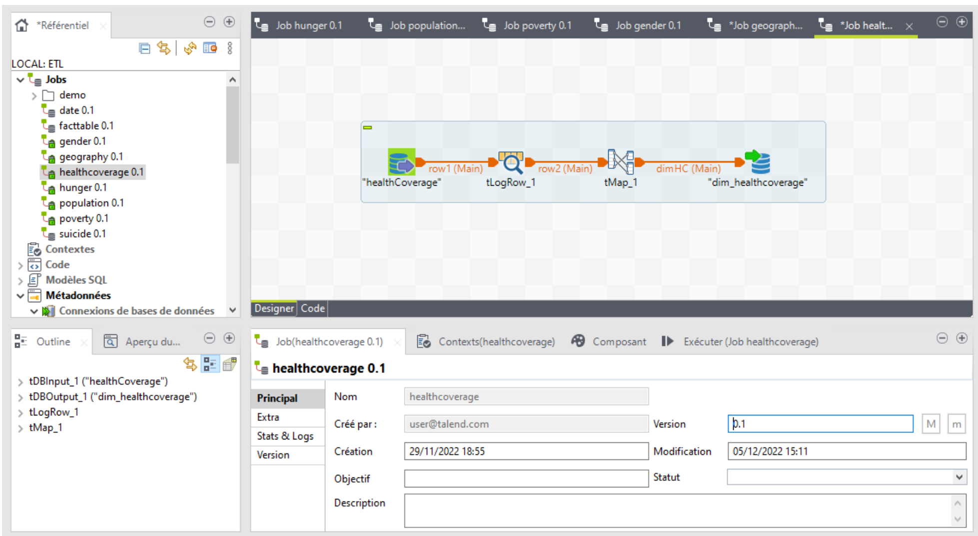
- *Gender Dimension*



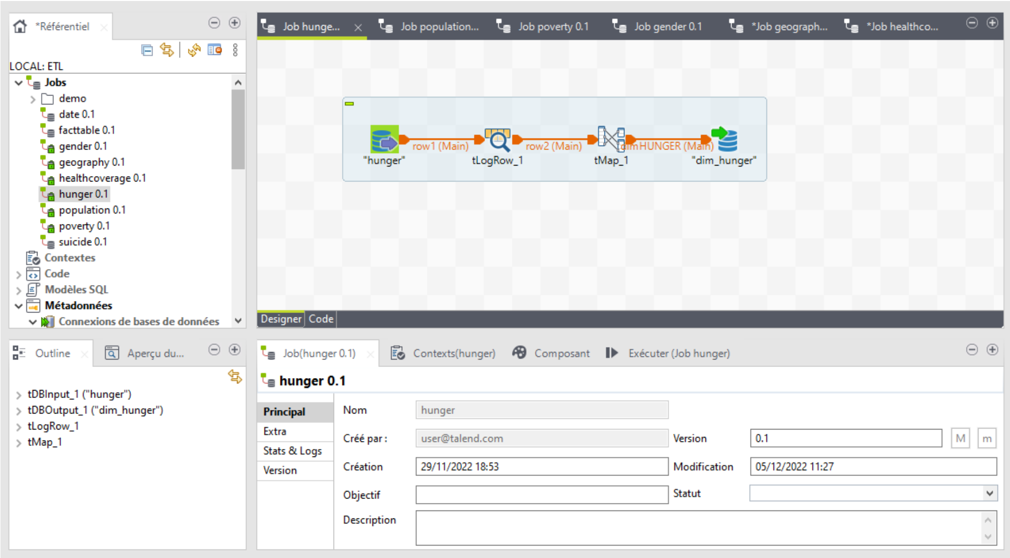


- *HealthCoverage Dimension*

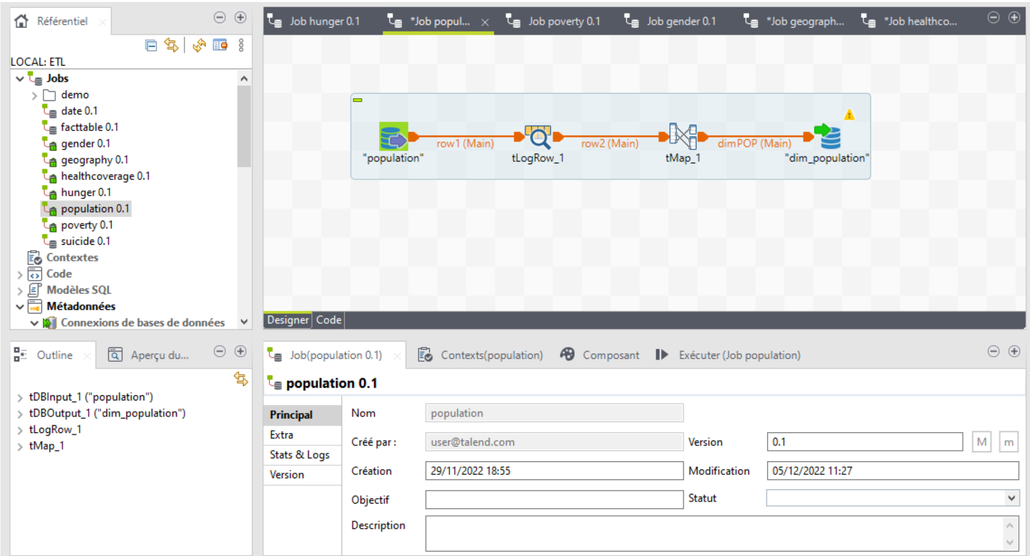
  the same applies to the others as regards the tMap



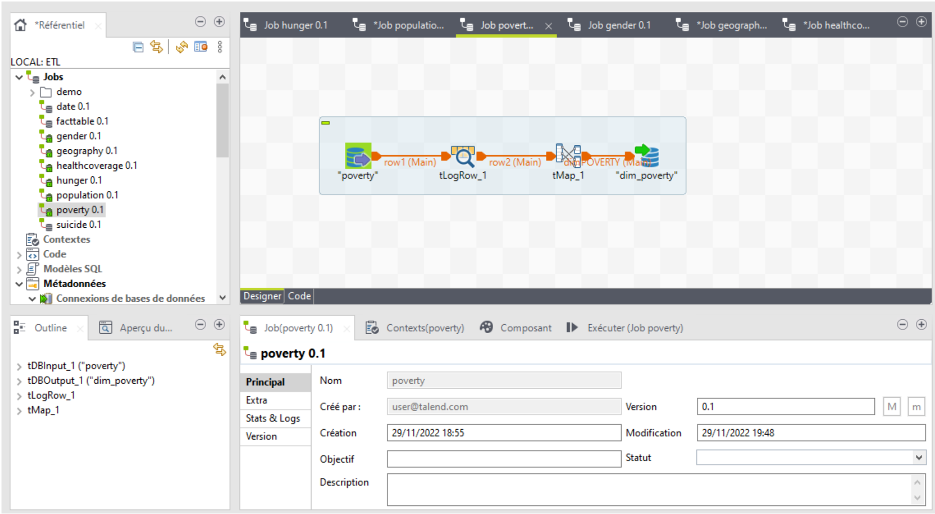
- *Hunger Dimension*



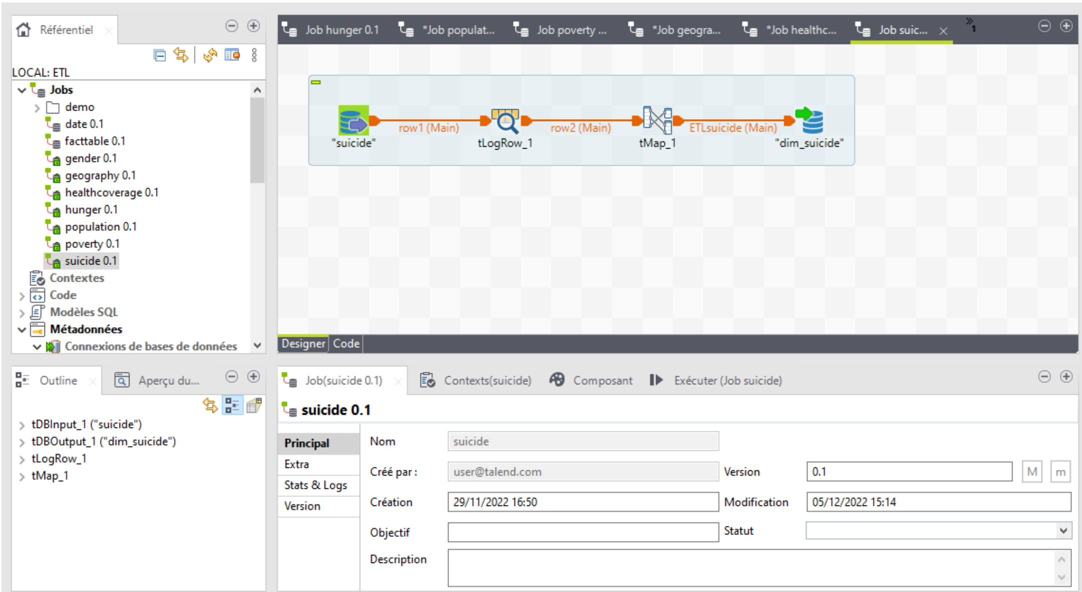
- *Population Dimension*



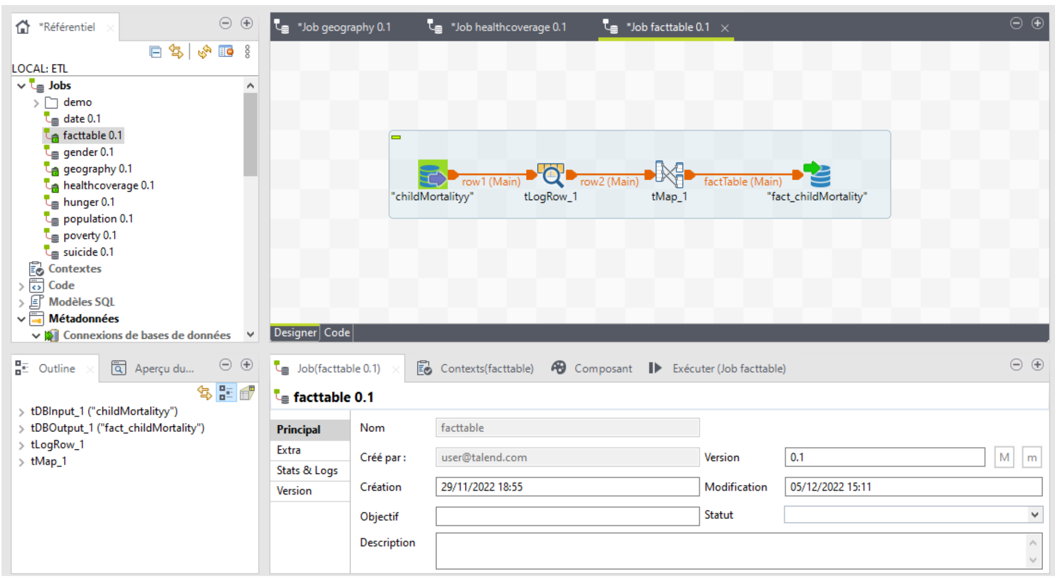
- *Poverty Dimension*



- *Suicide Dimension*

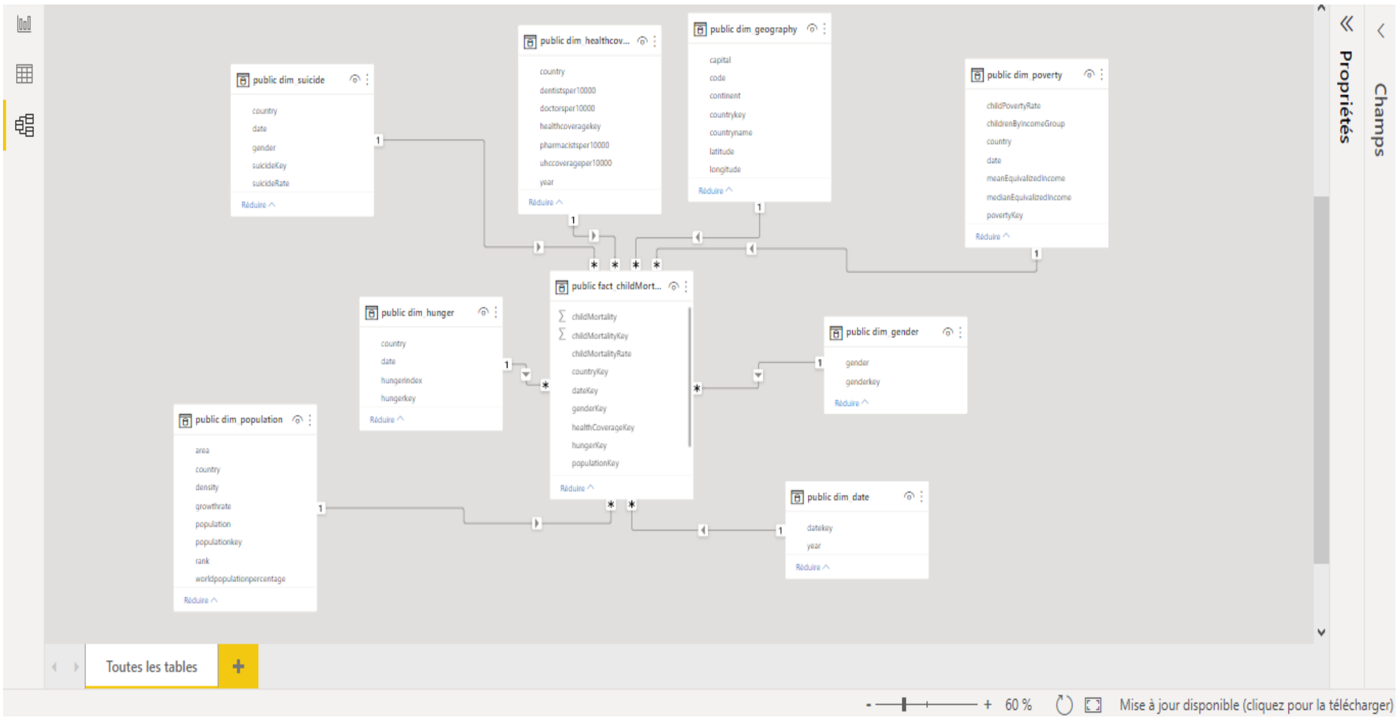


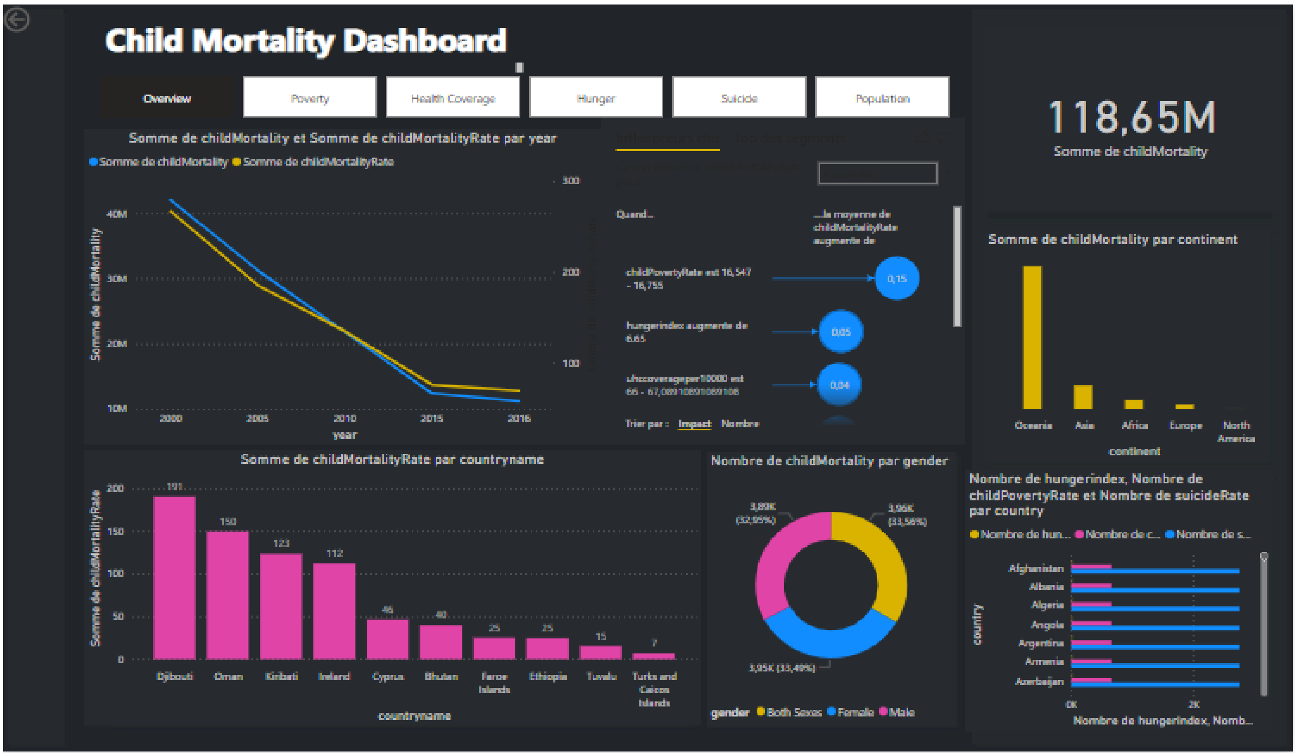
- *FactTable Dimension*



Reporting and Analysis

In this part, we will be constructing our analysis and dashboard using Microsoft Power BI, we will start first by establishing a link between Power BI and Talen and proceed to Import our data warehouse from the model tab:





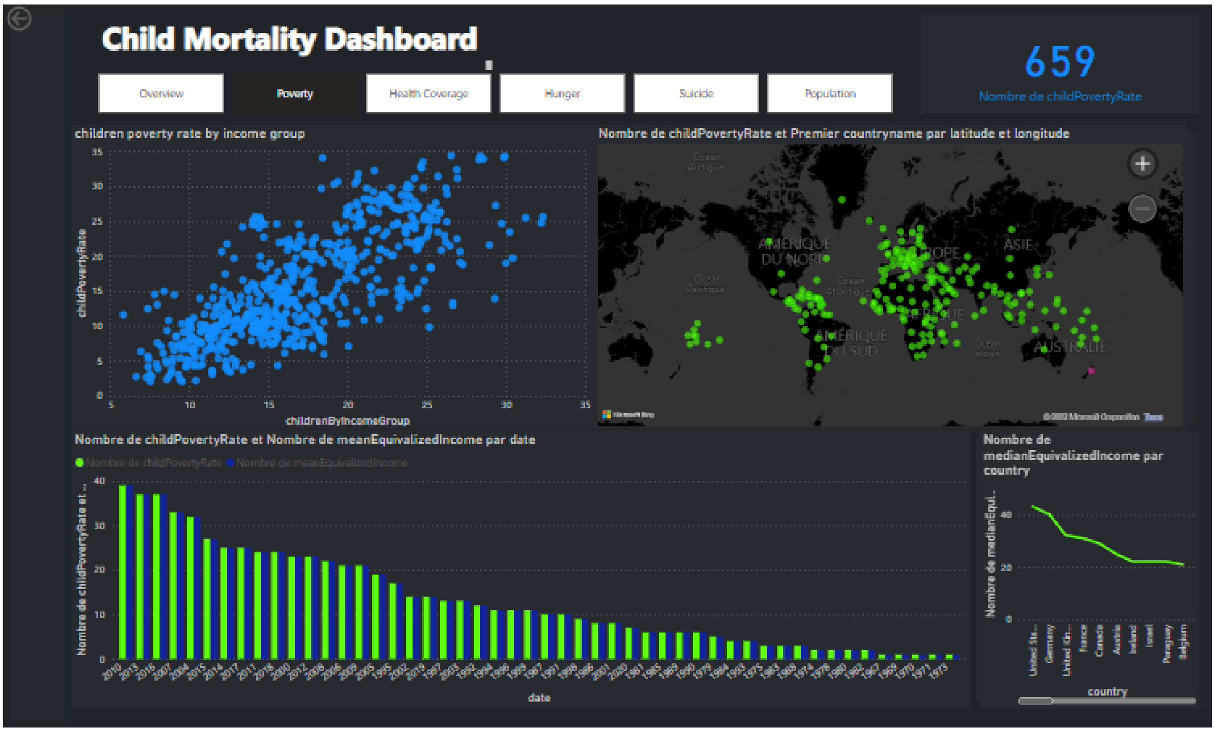
We proceed then to create graphics from our data warehouse on the rapport tab in addition to parameters if needed for a better visualization on our final dashboard.

Creation of the Dashboard

By creating a new dashboard tab, we can import the charts created later and insert them, we can also create filters to configure the display in an interactive way in our dashboard according to the dimensions

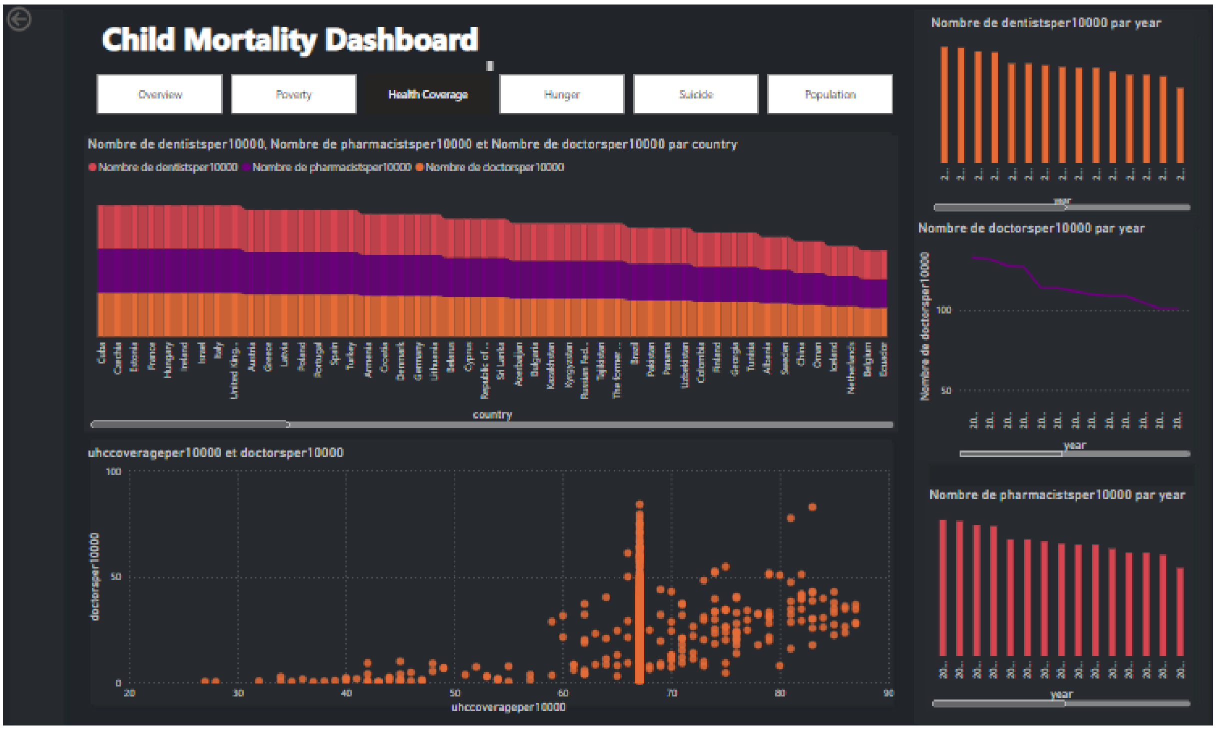
-Poverty Dimension:

this dashboard allows to have an overview on the impact of poverty on child mortality.



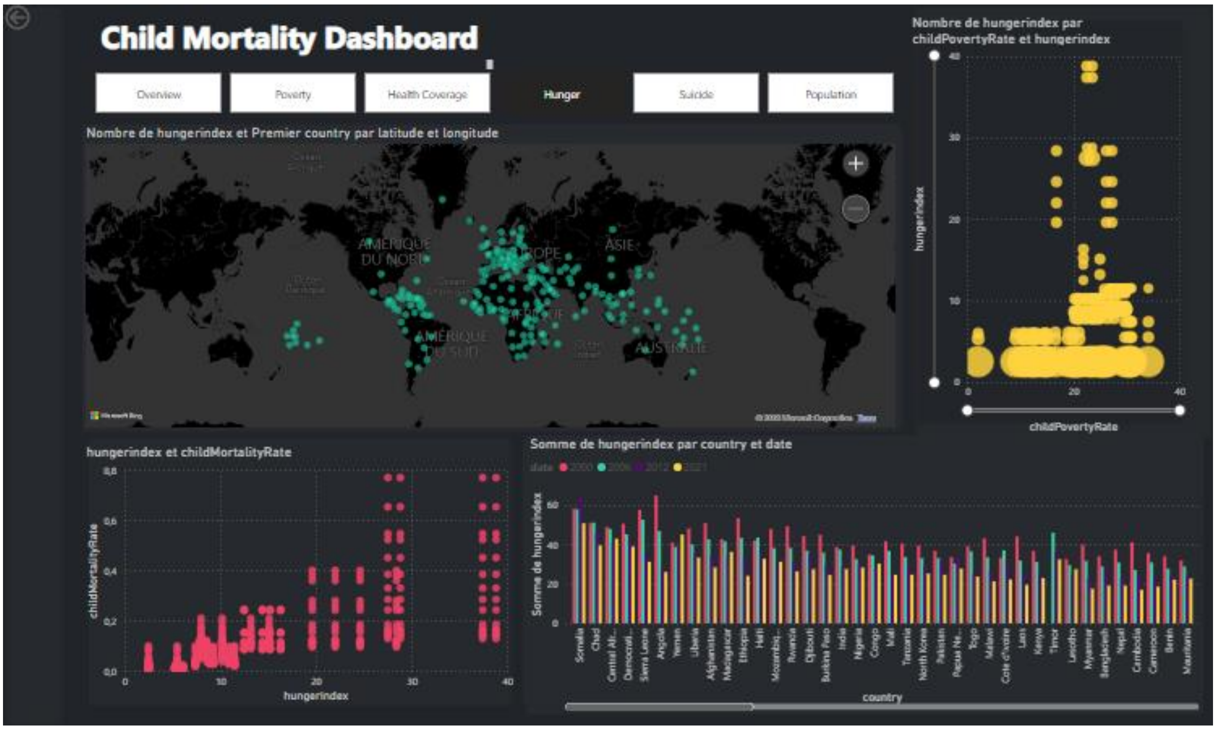
-Health Coverage Dimension:

this dashboard allows to have an overview on the impact of health coverage on child mortality.



-Hunger Dimension:

this dashboard allows to have an overview on the impact of hunger on child mortality.

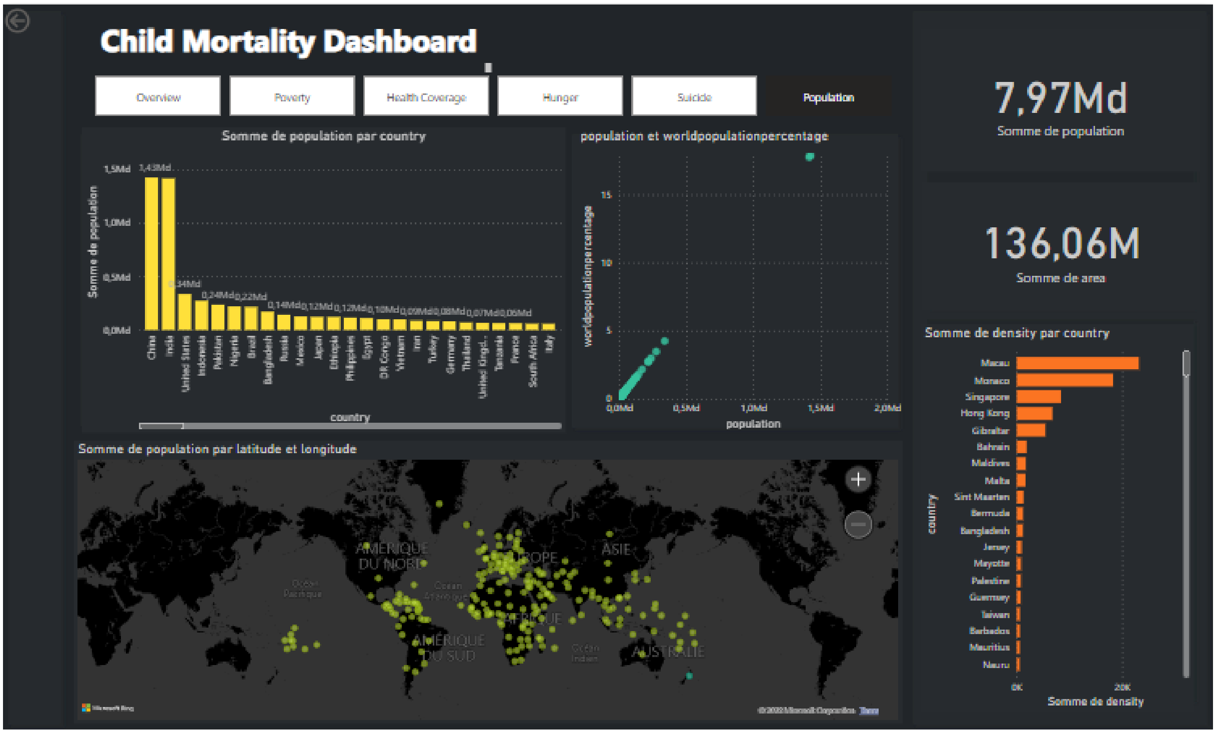


-Suicide Dimension:

this dashboard allows to have an overview on the impact of suicide on child mortality.

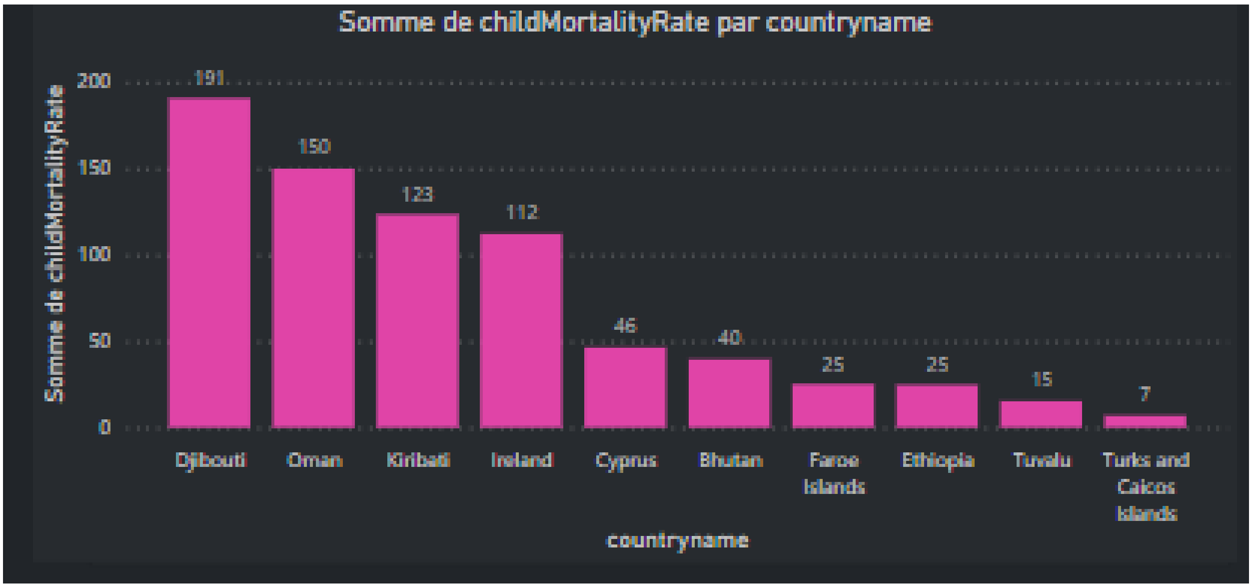
-Population Dimension:

this dashboard allows to have an overview on the impact of population on child mortality.



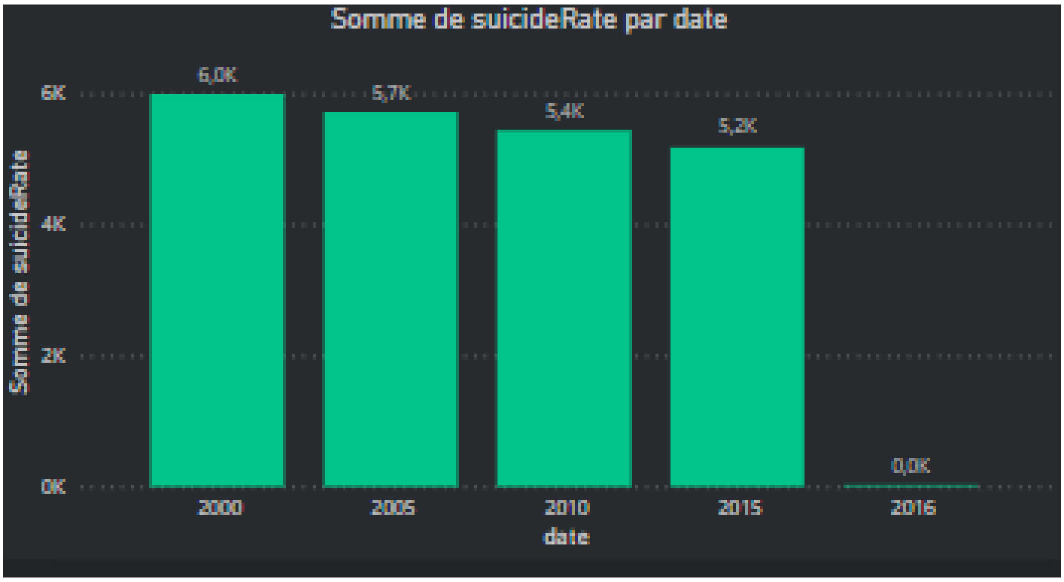
KPI Analysis:

1.Which country has the highest/lowest infant mortality rate?



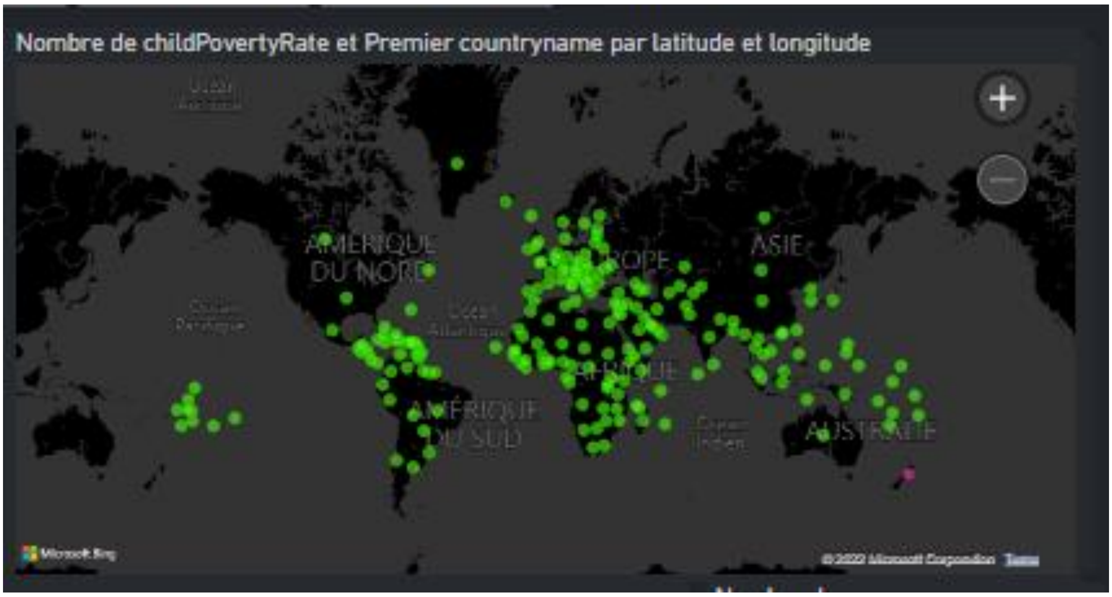
Analysis: This graph shows that the infant mortality rate is higher in Djibouti than in Turk and Cakos Islands

2. Which years had the highest/lowest child suicide rates?



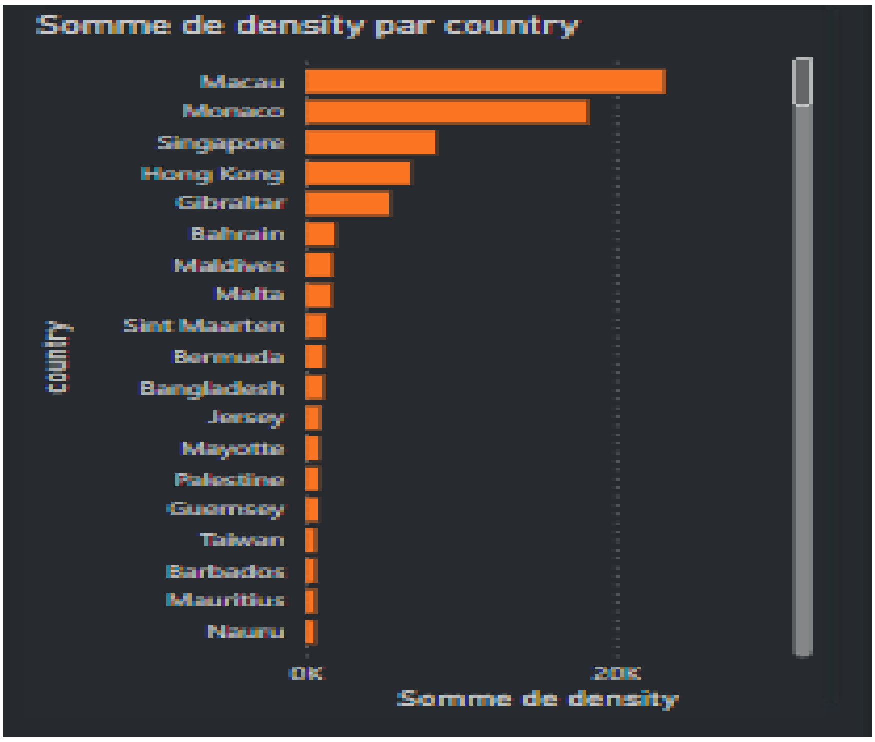
Analysis: The sum of the number of suicides decreased by year. In fact, we note that the year 2000 recognized the highest suicide rate

3. What is the impact of poverty on the distribution of infant mortality by latitude and longitude?



<u>Analysis:</u> Based on this analysis, it is observed that poverty has a great impact on the distribution of infant mortality in the map. Thus , we see that the African continent is the continent that has the most distribution point. So based on that, we can say that he suffers a lot of infant mortality because of poverty

4.what is the sum of density by country?



Analysis: According to this graph, Macau is the country with the highest population density compared to other countries.

Conclusion

as a concluding guide, the main purpose of this project that has been assigned to us is to carry out a case study in order to determine a relevant solution that allows us to reduce infant mortality in the world

Annex

Notebook about Data Collection

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

from IPython.display import FileLink

#load packages
import sys #access to system parameters https://docs.python.org/3/library/sys.html
print("Python version: {}". format(sys.version))

import pandas as pd #collection of functions for data processing and analysis modeled after R dat
aframes with SQL like features
print("pandas version: {}". format(pd.__version__))

import matplotlib.pyplot as plt #collection of functions for scientific and publication-ready vis
ualization

import numpy as np #foundational package for scientific computing
print("NumPy version: {}". format(np.__version__))

import scipy as sp #collection of functions for scientific computing and advance mathematics
print("SciPy version: {}". format(sp.__version__))

import seaborn as sns # for visualisation
print("Seaborn version: {}". format(sns.__version__))


#misc libraries
import random
import time
```

```
#misc libraries
import random
import time
```

```
/kaggle/input/datadw/SuicideData.csv
/kaggle/input/datadw/CountryData.csv
/kaggle/input/datadw/ChildMortalityRate.csv
/kaggle/input/datadw/gender.csv
/kaggle/input/suicide-rate-of-countries-per-every-year/suicideratefemale.csv
/kaggle/input/suicide-rate-of-countries-per-every-year/suicideratemale.csv
/kaggle/input/suicide-rate-of-countries-per-every-year/suicideratealll.csv
/kaggle/input/the-global-hunger-index/share-of-children-underweight.csv
/kaggle/input/the-global-hunger-index/global-hunger-index.csv
/kaggle/input/the-global-hunger-index/share-of-children-younger-than-5-who-suffer-from-stunt
ing.csv
/kaggle/input/the-global-hunger-index/share-of-children-with-a-weight-too-low-for-their-heig
ht-wasting.csv
/kaggle/input/database-files/SuicideData.csv
/kaggle/input/database-files/PovertyData.csv
/kaggle/input/database-files/CountryData.csv
/kaggle/input/database-files/PopulationData.csv
/kaggle/input/database-files/HungerData.csv
/kaggle/input/database-files/ChildMortalityRate.csv
/kaggle/input/database-files/HealthCoverageData.csv
/kaggle/input/global-child-mortality-rate/ChildMOrtalytRate.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/maternalMortalityRatio.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/neonatalMortalityRate.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/adolescentBirthRate.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/mortalityRateUnsafeWash.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/alcoholSubstanceAbuse.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/cleanFuelAndTech.csv
/kaggle/input/who-worldhealth-statistics-2020-complete/population10SDG3.8.2.csv
```

Child Mortality Rate File

In [2]:

```
ChildMortalityRate_dataRaw = pd.read_csv('../input/global-child-mortality-rate/ChildMOrtalytRat
e.csv')
```

In [3]:

```
ChildMortalityRate_dataRaw.head()
```

Out[3]:

	Unnamed: 0	Country	Year	Gender	Child Mortality(1 to 4)	Total Population	Mortality Rate
0	0	Afghanistan	1967	Female	26012.0	5080.813	5.119653
1	1	Afghanistan	1968	Female	26192.0	5202.606	5.034400
2	2	Afghanistan	1969	Female	26335.0	5333.936	4.937255
3	3	Afghanistan	1970	Female	26562.0	5476.630	4.850063
4	4	Afghanistan	1971	Female	26671.0	5630.099	4.737217

In [4]:

```
ChildMortalityRate_dataRaw.tail()
```

Out[4]:

	Unnamed: 0	Country	Year	Gender	Child Mortality(1 to 4)	Total Population	Mortality Rate
30935	30935	Zimbabwe	2015	Total	9031.0	13814.642	0.653727
30936	30936	Zimbabwe	2016	Total	8566.0	14030.338	0.610534
30937	30937	Zimbabwe	2017	Total	8318.0	14236.599	0.584269
30938	30938	Zimbabwe	2018	Total	7692.0	14438.812	0.532731
30939	30939	Zimbabwe	2019	Total	7397.0	14645.473	0.505071

Ta

In [5]:

```
ChildMortalityRate_dataRaw.shape
```

Out[5]:

```
(30940, 7)
```

In [6]:

```
ChildMortalityRate_dataRaw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30940 entries, 0 to 30939
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              30940 non-null  int64
 1   Country                 30940 non-null  object
 2   Year                    30940 non-null  int64
 3   Gender                  30940 non-null  object
 4   Child Mortality(1 to 4)  30940 non-null  float64
 5   Total Population        30064 non-null  float64
 6   Mortality Rate          30064 non-null  float64
dtypes: float64(3), int64(2), object(2)
memory usage: 1.7+ MB
```

In [7]:

```
ChildMortalityRate_dataRaw.rename(columns={'Unnamed: 0': 'ChildMortalityKey', 'Child Mortality
(1 to 4)' : 'ChildMortality'}, inplace=True)
```

In [8]:

```
ChildMortalityRate_dataRaw.drop_duplicates()
```

Out[8]:

	ChildMortalityKey	Country	Year	Gender	ChildMortality	Total Population	Mortality Rate
0	0	Afghanistan	1967	Female	26012.0	5080.813	5.119653
1	1	Afghanistan	1968	Female	26192.0	5202.606	5.034400
2	2	Afghanistan	1969	Female	26335.0	5333.936	4.937255
3	3	Afghanistan	1970	Female	26562.0	5476.630	4.850063
4	4	Afghanistan	1971	Female	26671.0	5630.099	4.737217
...	...	...	...	...	...	...	...
30935	30935	Zimbabwe	2015	Total	9031.0	13814.642	0.653727
30936	30936	Zimbabwe	2016	Total	8566.0	14030.338	0.610534
30937	30937	Zimbabwe	2017	Total	8318.0	14236.599	0.584269
30938	30938	Zimbabwe	2018	Total	7692.0	14438.812	0.532731
30939	30939	Zimbabwe	2019	Total	7397.0	14645.473	0.505071

30940 rows × 7 columns

In [9]:

```
missing_values_count = ChildMortalityRate_dataRaw.isnull().sum()
missing_values_count
```

Out[9]:

```
ChildMortalityKey      0
Country                0
Year                   0
Gender                 0
ChildMortality         0
Total Population     876
Mortality Rate       876
dtype: int64
```

In [10]:

```
# how many total missing values do we have?
total_cells = np.product(ChildMortalityRate_dataRaw.shape)
total_missing = missing_values_count.sum()

# percent of data that is missing
(total_missing/total_cells) * 100
```

Out[10]:

```
0.8089389601994644
```

we notice that a small percentage of our data has a NA value

In [11]:

```
ChildMortalityRate_dataRaw.describe()
```

Out[11]:

In [11]:

```
ChildMortalityRate_dataRaw.describe()
```

Out[11]:

	ChildMortalityKey	Year	ChildMortality	Total Population	Mortality Rate
count	30940.000000	30940.000000	3.094000e+04	3.006400e+04	30064.000000
mean	15469.500000	1991.456561	1.272722e+04	1.975113e+04	0.959470
std	8931.753001	17.323382	6.370284e+04	8.053780e+04	1.481062
min	0.000000	1955.000000	0.000000e+00	1.606000e+00	0.000000
25%	7734.750000	1978.000000	6.900000e+01	9.928217e+02	0.044134
50%	15469.500000	1993.000000	6.490000e+02	3.890678e+03	0.225487
75%	23204.250000	2006.000000	6.499500e+03	1.175135e+04	1.292107
max	30939.000000	2019.000000	1.463821e+06	1.433784e+06	10.878031

In [12]:

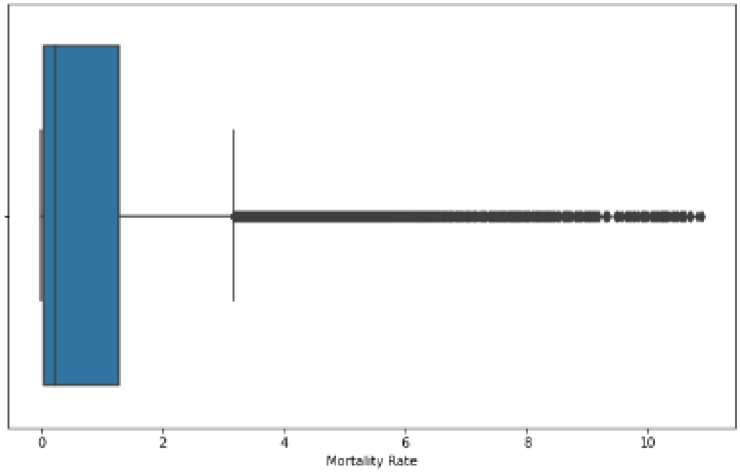
```
fig, ax = plt.subplots(figsize=(10, 6))
sns.boxplot(ChildMortalityRate_dataRaw['Mortality Rate'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the fo
llowing variable as a keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will result in an er
ror or misinterpretation.
  FutureWarning
```

Out[12]:

Out[12]:

```
<AxesSubplot:xlabel='Mortality Rate'>
```



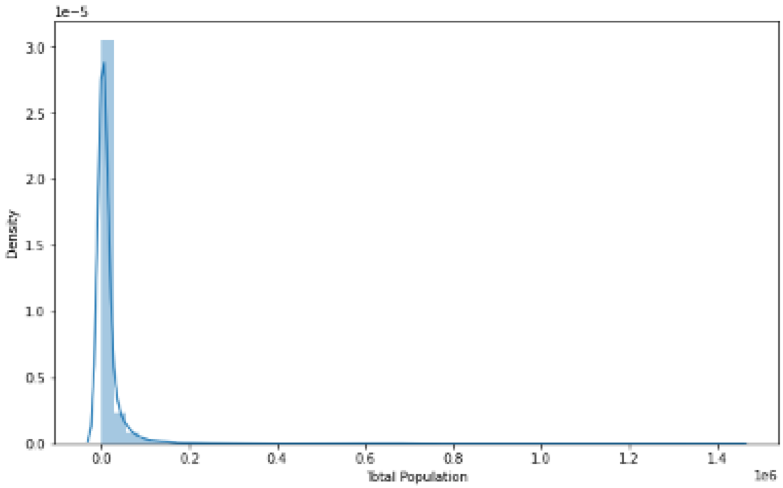
In [13]:

```
fig, ax = plt.subplots(figsize=(10, 6))
sns.distplot(ChildMortalityRate_dataRaw['Total Population'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distpl
ot` is a deprecated function and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:

```
<AxesSubplot:xlabel='Total Population', ylabel='Density'>
```



Replacing missing values

In [14]:

```
ChildMortalityRate_dataRaw['Total Population'] = ChildMortalityRate_dataRaw['Total Populatio
n'].fillna(ChildMortalityRate_dataRaw['Total Population'].mean())
ChildMortalityRate_dataRaw['Mortality Rate'] = ChildMortalityRate_dataRaw['Mortality Rate'].fil
lna(ChildMortalityRate_dataRaw['Mortality Rate'].mean())
```

In [15]:

```
ChildMortalityRate_dataRaw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30940 entries, 0 to 30939
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ChildMortalityKey  30940 non-null  int64
 1   Country            30940 non-null  object
 2   Year               30940 non-null  int64
 3   Gender             30940 non-null  object
 4   ChildMortality     30940 non-null  float64
 5   Total Population   30940 non-null  float64
 6   Mortality Rate     30940 non-null  float64
dtypes: float64(3), int64(2), object(2)
memory usage: 1.7+ MB
```

In [16]:

```
ChildMortalityRate_dataRaw.rename(columns={'Total Population': 'TotalPopulation', 'Mortality Ra
te' : 'MortalityRate'}, inplace=True)
```

In [17]:

```
ChildMortalityRate_dataRaw.head()
```

Out[17]:

	ChildMortalityKey	Country	Year	Gender	ChildMortality	TotalPopulation	MortalityRate
0	0	Afghanistan	1967	Female	26012.0	5080.813	5.119653
1	1	Afghanistan	1968	Female	26192.0	5202.606	5.034400
2	2	Afghanistan	1969	Female	26335.0	5333.936	4.937255
3	3	Afghanistan	1970	Female	26562.0	5476.630	4.850063
4	4	Afghanistan	1971	Female	26671.0	5630.099	4.737217

In [18]:

```
ChildMortalityRate_dataRaw.dtypes
```

Out[18]:

```
ChildMortalityKey      int64
Country               object
Year                   int64
Gender                object
ChildMortality       float64
TotalPopulation      float64
MortalityRate        float64
dtype: object
```

In [19]:

```
ChildMortalityRate_dataRaw.drop('ChildMortalityKey', inplace=True, axis=1)
```

In [20]:

```
def add_id_column(df, columnName):
    df.insert(0, columnName, (df.index)+1)
```


In [20]:

```
def add_id_column(df, columnName):
    df.insert(0, columnName, (df.index)+1)
```


In [21]:

```
add_id_column(ChildMortalityRate_dataRaw, 'ChildMortalityID')
```

In [22]:

```
# Generate a csv file
ChildMortalityRate_dataRaw.to_csv('ChildMortalityRate.csv', encoding='utf-8', index=False)
```

This file is now ready !!

In [23]:

```
# let's import it to our output space
import os
os.chdir(r'/kaggle/working')
FileLink(r'ChildMortalityRate.csv')
```

Out[23]:

ChildMortalityRate.csv

Country Data File

In [24]:

```
CountryData = pd.read_csv('../input/counties-geographic-coordinates/countries.csv')
```

In [25]:

```
CountryData.head()
```

Out[25]:

	country	latitude	longitude	name
0	AD	42.546245	1.601554	Andorra
1	AE	23.424076	53.847818	United Arab Emirates
2	AF	33.939110	67.709953	Afghanistan
3	AG	17.060816	-61.796428	Antigua and Barbuda
4	AI	18.220554	-63.068615	Anguilla

In [26]:

```
CountryData.tail()
```

Out[26]:

	country	latitude	longitude	name
240	YE	15.552727	48.516388	Yemen
241	YT	-12.827500	45.166244	Mayotte
242	ZA	-30.559482	22.937506	South Africa
243	ZM	-13.133897	27.849332	Zambia
244	ZW	-19.015438	29.154857	Zimbabwe

See more in the link of the notebook:

https://www.kaggle.com/code/ouissalmifdal/data-preparation