ECBS 5146 SQL and Different Shapes of Data

Term 2 Project

Hasan Mansoor Khan, Ahmed Mohammad, Gabriella Zsiros

Contents

1. Choice of Dataset

One of the primary tasks we performed as a group was to select a suitable dataset for the purpose of analysis. After some fruitful deliberations we opted for a migration dataset from Eurostat. The dataset was downloaded from Eurostat in CSV format & uploaded on the MySQL Workbench. We then proceeded to import the dataset in a KNIME workflow.

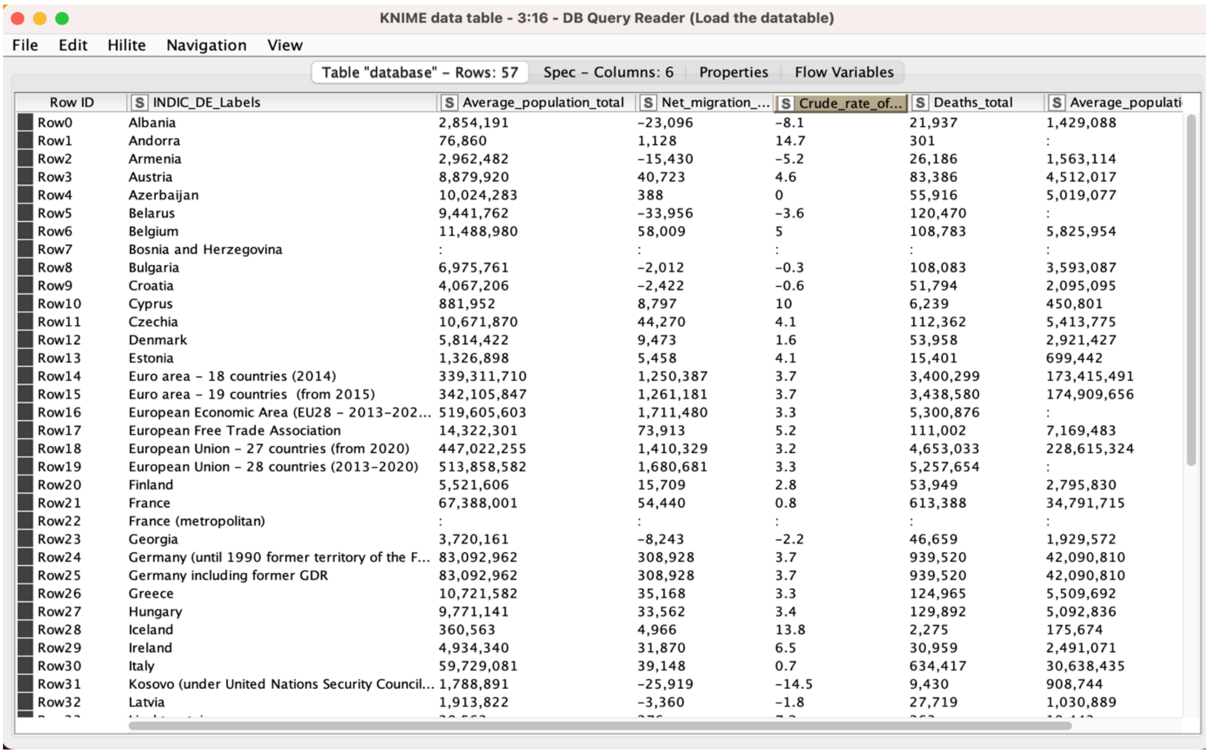
The migration dataset from Eurostat had many important variables such as the net migration in absolute terms, the average population of the country, the crude rate of migration, the deaths in absolute terms as well as the average female population in a country. As far as observations are concerned, each row represents a country for which Eurostat collects data. The vast majority are EU countries, but the data also represents important countries not part of the EU, such as the United Kingdom, and Tukey. Moreover, the dataset also includes main aggregations including the Euro area as well as the European Economic Area which each include 19 and 28 countries respectively. Another important measure is the time which is measured in years. For the purpose of this analysis, we used Eurostat's built-in filter for extracting the dataset for 2019 values. The reason is that primarily we aimed for a cross-sectional analysis rather than a time series. The logic behind selecting 2019 is because 2019 is the most recent year considering that 2020 and 2021 were outliers for migration due to the Covid-19 pandemic. Hence, our analysis of the dataset rests on multiple migration variables for countries in 2019.

The selection of the datasets was a consensus within the team, with Hasan importing it into SQL and major part of the documentation, Ahmed cleaning the of the data and major part of the documentation, Gabriella bringing in the two datasets with API and File loader, initial sketch of the workflow and editing of the documentation. The general outcome and the final shape of the workflow and visualization was a joint effort with a lot of meetings and discussions.

1.1. Details of each variable

1. **Row ID:** Unique row id autogenerated in KNIME workflow
2. **INDIC_DE_Labels:** Represents the name of the country for which the data is collected.
3. Average Population Total: The average population of the given country throughout the year 2019
4. Net Migration: The net flow of migrants calculated by deducting Emigrants from Immigrants. Hence a positive net migration is achieved when more people are immigrating than they are emigrating.
5. Crude Rate of Net Migration: This is the consequential variable which measures net migration as ratio of average population. This incorporates the different population of each country & their respective migration numbers. This variable is essential for making comparisons between especially, especially between countries with varying populations.
6. Deaths Total: The total number of deaths in a country.
7. Average population of Females: The number of females within the population of a country.

The dataset is in a table format as seen below:


KNIME data table - 3:16 - DB Query Reader (Load the datatable)
File
Edit
Hilite
Navigation
View
Table "database" – Rows: 57
Spec – Columns: 6
Properties
Flow Variables

Row ID	S INDIC_DE_Labels	S Average_population_total	S Net_migration_...	S Crude_rate_of...	S Deaths_total	S Average_populati
Row0	Albania	2,854,191	-23,096	-8.1	21,937	1,429,088
Row1	Andorra	76,860	1,128	14.7	301	:
Row2	Armenia	2,962,482	-15,430	-5.2	26,186	1,563,114
Row3	Austria	8,879,920	40,723	4.6	83,386	4,512,017
Row4	Azerbaijan	10,024,283	388	0	55,916	5,019,077
Row5	Belarus	9,441,762	-33,956	-3.6	120,470	:
Row6	Belgium	11,488,980	58,009	5	108,783	5,825,954
Row7	Bosnia and Herzegovina	:	:	:	:	:
Row8	Bulgaria	6,975,761	-2,012	-0.3	108,083	3,593,087
Row9	Croatia	4,067,206	-2,422	-0.6	51,794	2,095,095
Row10	Cyprus	881,952	8,797	10	6,239	450,801
Row11	Czechia	10,671,870	44,270	4.1	112,362	5,413,775
Row12	Denmark	5,814,422	9,473	1.6	53,958	2,921,427
Row13	Estonia	1,326,898	5,458	4.1	15,401	699,442
Row14	Euro area – 18 countries (2014)	339,311,710	1,250,387	3.7	3,400,299	173,415,491
Row15	Euro area – 19 countries (from 2015)	342,105,847	1,261,181	3.7	3,438,580	174,909,656
Row16	European Economic Area (EU28 – 2013-202...	519,605,603	1,711,480	3.3	5,300,876	:
Row17	European Free Trade Association	14,322,301	73,913	5.2	111,002	7,169,483
Row18	European Union – 27 countries (from 2020)	447,022,255	1,410,329	3.2	4,653,033	228,615,324
Row19	European Union – 28 countries (2013-2020)	513,858,582	1,680,681	3.3	5,257,654	:
Row20	Finland	5,521,606	15,709	2.8	53,949	2,795,830
Row21	France	67,388,001	54,440	0.8	613,388	34,791,715
Row22	France (metropolitan)	:	:	:	:	:
Row23	Georgia	3,720,161	-8,243	-2.2	46,659	1,929,572
Row24	Germany (until 1990 former territory of the F...	83,092,962	308,928	3.7	939,520	42,090,810
Row25	Germany including former GDR	83,092,962	308,928	3.7	939,520	42,090,810
Row26	Greece	10,721,582	35,168	3.3	124,965	5,509,692
Row27	Hungary	9,771,141	33,562	3.4	129,892	5,092,836
Row28	Iceland	360,563	4,966	13.8	2,275	175,674
Row29	Ireland	4,934,340	31,870	6.5	30,959	2,491,071
Row30	Italy	59,729,081	39,148	0.7	634,417	30,638,435
Row31	Kosovo (under United Nations Security Council...	1,788,891	-25,919	-14.5	9,430	908,744
Row32	Latvia	1,913,822	-3,360	-1.8	27,719	1,030,889

1.2. Prerequisite for replicating and running the workflow

The SQL database needs to be created in MySQL, as the KNIME workflow uses it as a source. The provided .sql script creates the scheme “term2” with the database “e3”, which is then imported with the MySQL connector. The other two data sources are from cloud server and API query.

2. Importing data

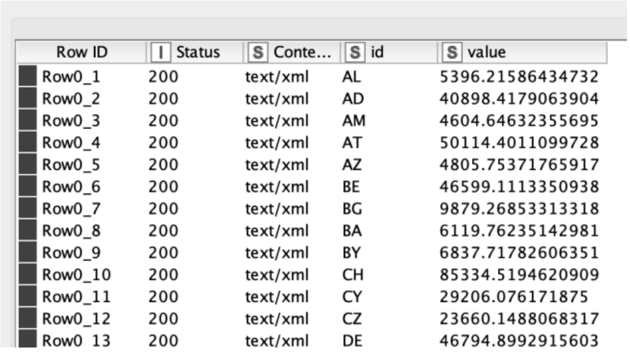
2.1. MySQL

Using the MySQL Connector Node we connect our SQL with KNIME using our SQL credentials. We use the DB Query Reader Node to read the specific database and table. For this analysis we imported the term 2 SQL database and the table e3 which is the migration dataset from Eurostat.

https://ec.europa.eu/eurostat/databrowser/view/demo_gind/default/table?lang=en

2.2. Importing data from an API

Apart from the Eurostat data, an API matched data was imported using the file reader node. For this we used the World Bank API. The data we chose to import from this API is the GDP per capita data which is loaded from World Bank database (using the relevant indicator *NY.GDP.PCAP.CD*). The GDP per capita is an integral measure because it takes into account the population of a country when measuring GDP.

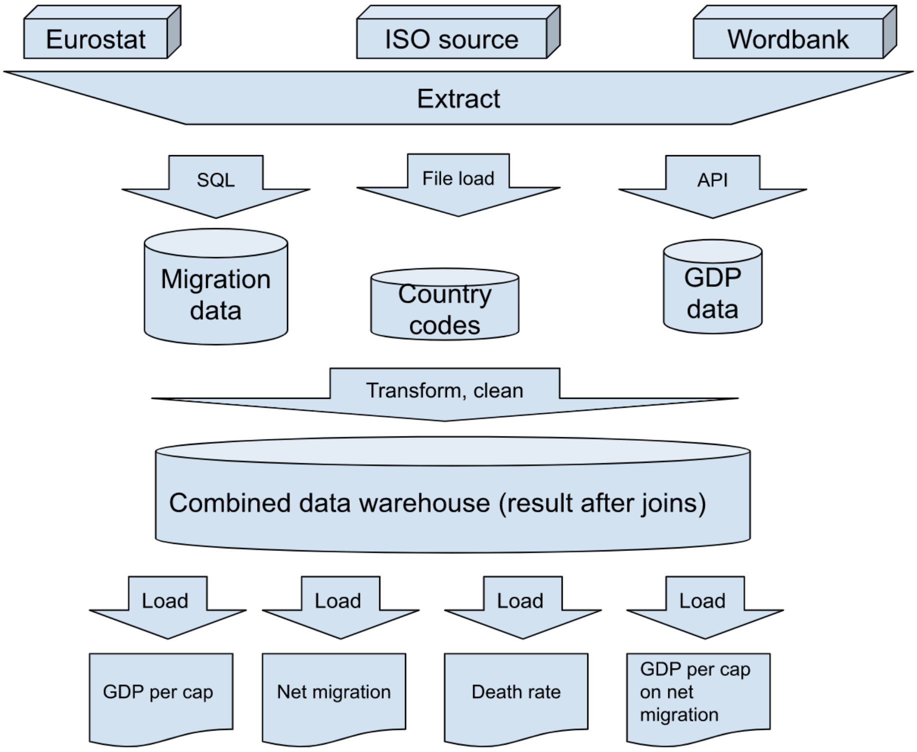


Row ID	I Status	S Conte...	S id	S value
Row0_1	200	text/xml	AL	5396.21586434732
Row0_2	200	text/xml	AD	40898.4179063904
Row0_3	200	text/xml	AM	4604.64632355695
Row0_4	200	text/xml	AT	50114.4011099728
Row0_5	200	text/xml	AZ	4805.75371765917
Row0_6	200	text/xml	BE	46599.1113350938
Row0_7	200	text/xml	BG	9879.26853313318
Row0_8	200	text/xml	BA	6119.76235142981
Row0_9	200	text/xml	BY	6837.71782606351
Row0_10	200	text/xml	CH	85334.5194620909
Row0_11	200	text/xml	CY	29206.076171875
Row0_12	200	text/xml	CZ	23660.1488068317
Row0_13	200	text/xml	DE	46794.8992915603

2.3. File loader

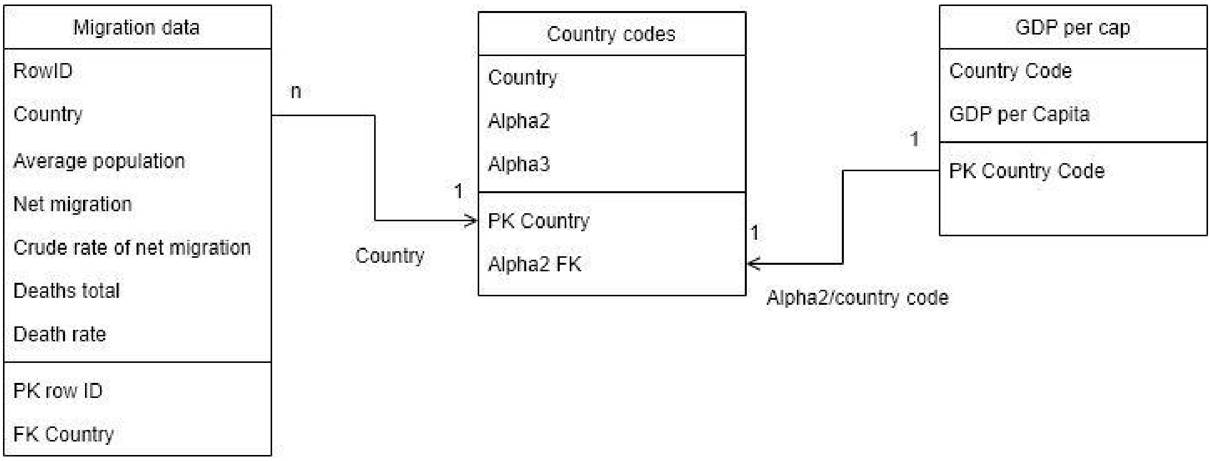
Moreover, using the World Bank API we also imported the Country codes file. Given the difference in naming conventions of countries in the two datasets, a safer choice was to introduce an auxiliary table, which takes conventional country names and ISO 3166 classification of country code - both 2- and 3-character versions so it can be later evaluated which one we need. These country codes allow for being used in visualizations as well as identifying the country in a need table using their well-known country codes.

3. Data Structure & ETL



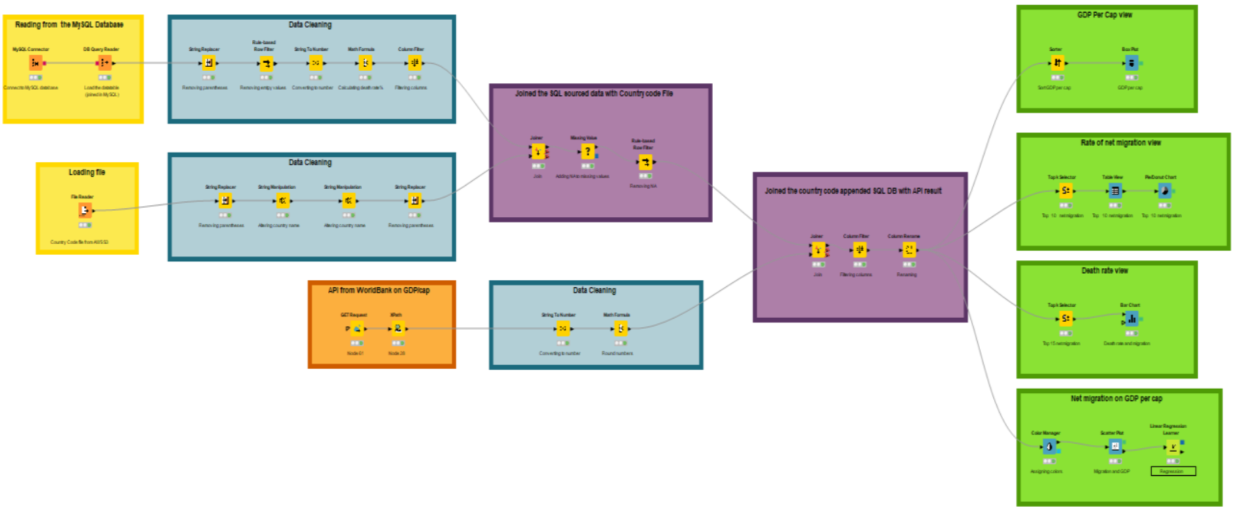
The above chart captures the Extract, Transform & Load pipeline of our project. Using SQL, files on AWS S3 & the World Bank API we load Migration data, country codes and GDP per capita. After data cleaning, we proceed to joining the data in a combined data warehouse after joining the three cleaned data tables. Lastly, we load the four visualizations using our combined data warehouse as the input data.

3.1. ER Diagram



The ER diagram shows the 3 data sources and the connections made. The first connection is made between the migration data from Eurostat and the Country Codes from the AWS S3 server which is made by the Country in migration table and PK Country in the country codes table. This data is then connected with the third data source which is called using an API. The connection is made by Alpha2 Foreign Key and PK Country Code. This allows creation of a relational database using unique identifiers using joins.

4. KNIME Workflow



The final KNIME workflow is created by first importing all 3 datasets, then perform data cleaning, then joining and lastly visualizing. The Yellow and Orange colors highlight those nodes that import the data while the blue ones represent data cleaning. The purple highlighted nodes represent the nodes that perform the joins while the green section of workflow shows the visualizations.

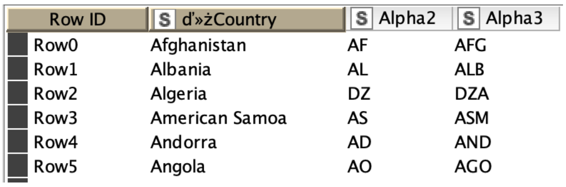
4.1. ETL in KNIME

This section contains detailed information regarding all the nodes that were used in the project along with the reason behind selecting that node, the configurations that were done inside it and how the results obtained were useful for our analysis.

4.1.1. Reading the dataset

The first workflow annotation includes two nodes and is the initiation point of our ETL pipeline in KNIME. The first step is to connect MySQL to KNIME which was done using the MySQL connector node. This node requires Hostname, database name, port, and login credentials to successfully connect to MySQL. Once this is done, we loaded the data table that was joined in SQL workbench into KNIME through DB Query Reader node. Adding a snippet of our data table that was loaded into KNIME. This table contains missing values, unwanted columns and misleading headers which will all be accounted for in the coming steps.

In addition to these nodes, we also used File Reader node to upload the Country Codes that were obtained through a file that was uploaded on Amazon S3 in a csv file. The URL was provided, and all Country Codes were fetched using this node. The idea is to include these country codes in our main data table along with country names. An image has been attached as reference to understand the contents of this table.



Row ID	S d'»żCountry	S Alpha2	S Alpha3
Row0	Afghanistan	AF	AFG
Row1	Albania	AL	ALB
Row2	Algeria	DZ	DZA
Row3	American Samoa	AS	ASM
Row4	Andorra	AD	AND
Row5	Angola	AO	AGO

We will be performing data cleaning on the table obtained from DB Query Reader and File Reader separately.

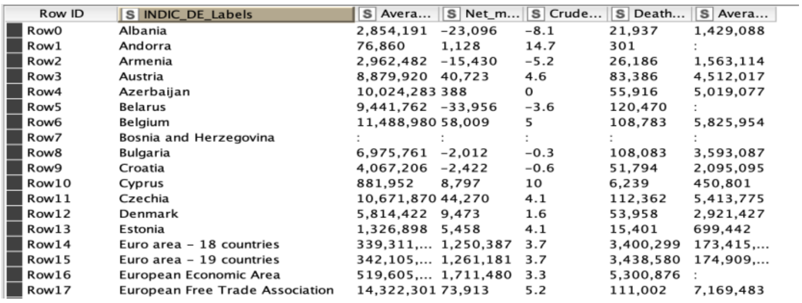
The World Bank data is loaded via an API into the KNIME workflow using the GET Request Node and the XPath nodes. These nodes allow the GDP per capita to be imported and can be later joined with the remaining dataset after cleaning. The nodes are configured with the World Bank API which is as follows:

http://api.worldbank.org/v2/country/AL;AD;AM;AT;AZ;BY;BE;BA;BG;HR;CY;CZ;DK;EE;FI;FR;GE;DE;GR;HU;IS;IE;IT;LV;LI;LT;LU;MT;MC;ME;NL;MK;NO;PL;PT;RO;SM;RS;SK;SI;ES;SE;CH;TR;UA;GB/indicator/NY.GDP.PCAP.CD?date=2019&format=XML&compressed=FALSE

4.1.2. Cleaning the dataset

Step 1: Cleaning data from DB Query Reader:

Looking at the results, we notice that country names have parentheses in the observations which need to be cleaned. To get clean names, we aim to remove the text given in brackets and have seamless values for our Country Name column. We used the String Replacer node which allowed us to configure a wildcard pattern and extract only the names of countries, disregarding the text in parentheses. As it can be seen in the image attached below, our objective to remove the parentheses has been achieved.



Row ID	S INDIC_DE_Labels	S Avera...	S Net_m...	S Crude...	S Death...	S Avera...
Row0	Albania	2,854,191	-23,096	-8.1	21,937	1,429,088
Row1	Andorra	76,860	1,128	14.7	301	:
Row2	Armenia	2,962,482	-15,430	-5.2	26,186	1,563,114
Row3	Austria	8,879,920	40,723	4.6	83,386	4,512,017
Row4	Azerbaijan	10,024,283	388	0	55,916	5,019,077
Row5	Belarus	9,441,762	-33,956	-3.6	120,470	:
Row6	Belgium	11,488,980	58,009	5	108,783	5,825,954
Row7	Bosnia and Herzegovina	:	:	:	:	:
Row8	Bulgaria	6,975,761	-2,012	-0.3	108,083	3,593,087
Row9	Croatia	4,067,206	-2,422	-0.6	51,794	2,095,095
Row10	Cyprus	881,952	8,797	10	6,239	450,801
Row11	Czechia	10,671,870	44,270	4.1	112,362	5,413,775
Row12	Denmark	5,814,422	9,473	1.6	53,958	2,921,427
Row13	Estonia	1,326,898	5,458	4.1	15,401	699,442
Row14	Euro area – 18 countries	339,311,...	1,250,387	3.7	3,400,299	173,415,...
Row15	Euro area – 19 countries	342,105,...	1,261,181	3.7	3,438,580	174,909,...
Row16	European Economic Area	519,605,...	1,711,480	3.3	5,300,876	:
Row17	European Free Trade Association	14,322,301	73,913	5.2	111,002	7,169,483

Step 2: Removing empty values from our data-table

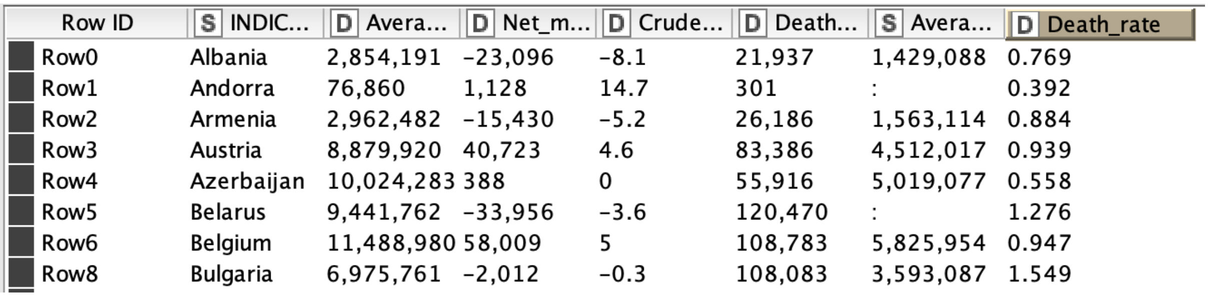
For this step, we used a Rule-based Row Level Filter node which was configured to remove the empty observations from our data. This node looked at our input data and checked for empty values in the Average Population column. Any row which returned a TRUE value was dropped from the data, optimizing it, and making it more coherent.

Step 3: Converting string values to numbers

Multiple columns in our data table were returning values in the form of strings. For our analysis, we required values as an integer, so we used the String to Number node to change the data type.

Step 4: Calculation of death-rate percentage:

Once we had our desired columns successfully converted to integer values, we performed a calculation in order to create a new column that gave us a result of total deaths divided by the total population. For this, we used a node called Math Formula and obtained a new calculated column as in the image below.



Row ID	S INDIC...	D Avera...	D Net_m...	D Crude...	D Death...	S Avera...	D Death_rate
Row0	Albania	2,854,191	-23,096	-8.1	21,937	1,429,088	0.769
Row1	Andorra	76,860	1,128	14.7	301	:	0.392
Row2	Armenia	2,962,482	-15,430	-5.2	26,186	1,563,114	0.884
Row3	Austria	8,879,920	40,723	4.6	83,386	4,512,017	0.939
Row4	Azerbaijan	10,024,283	388	0	55,916	5,019,077	0.558
Row5	Belarus	9,441,762	-33,956	-3.6	120,470	:	1.276
Row6	Belgium	11,488,980	58,009	5	108,783	5,825,954	0.947
Row8	Bulgaria	6,975,761	-2,012	-0.3	108,083	3,593,087	1.549

Step 5: Filtering columns based on the requirements of our analysis

In the final step of data cleaning for this table, we wanted to drop any column that wasn't deemed necessary for our analysis. The node used for this step was Column Filter node. Average_population_females was present in our data table but was dropped eventually.

Moving on to the second Data Cleaning process for the Country Codes table obtained from File Reader:

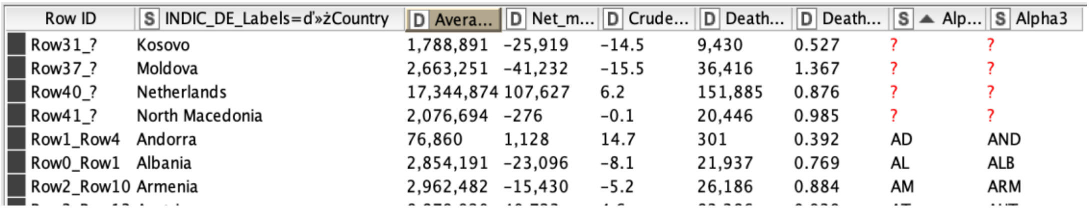
Step 1: Removing parenthesis from Country Name column:

As we performed this step in Data Cleaning earlier, we used the String Replacer node again to remove any parentheses present in the observations for Country name which was done successfully.

Step 2: Altering country names:
As we noticed a few countries in our data which had their names updated according to the latest country name list, we altered their names by attaching two String Manipulation nodes. Turkey and Macedonia were changed to Türkiye and North Macedonia respectively. In addition to these nodes, a String Replacer node was added which allowed us to change ‘The United Kingdom’ to ‘United Kingdom’. This concluded our data cleaning process for the table obtained from File Reader.

4.1.3. Joining the two data tables

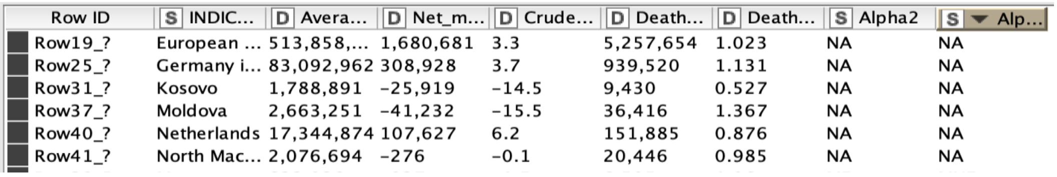
Once both our data tables were cleaned and we achieved the desired output which had the country values in sync with each other, we used a Joiner node to combine the two tables into one table that had all our data. Here is the output obtained from our Joiner node:



Row ID	S INDIC_DE_Labels=d’»żCountry	D Avera...	D Net_m...	D Crude...	D Death...	D Death...	S ▲ Alp...	S Alpha3
Row31_?	Kosovo	1,788,891	-25,919	-14.5	9,430	0.527	?	?
Row37_?	Moldova	2,663,251	-41,232	-15.5	36,416	1.367	?	?
Row40_?	Netherlands	17,344,874	107,627	6.2	151,885	0.876	?	?
Row41_?	North Macedonia	2,076,694	-276	-0.1	20,446	0.985	?	?
Row1_Row4	Andorra	76,860	1,128	14.7	301	0.392	AD	AND
Row0_Row1	Albania	2,854,191	-23,096	-8.1	21,937	0.769	AL	ALB
Row2_Row10	Armenia	2,962,482	-15,430	-5.2	26,186	0.884	AM	ARM

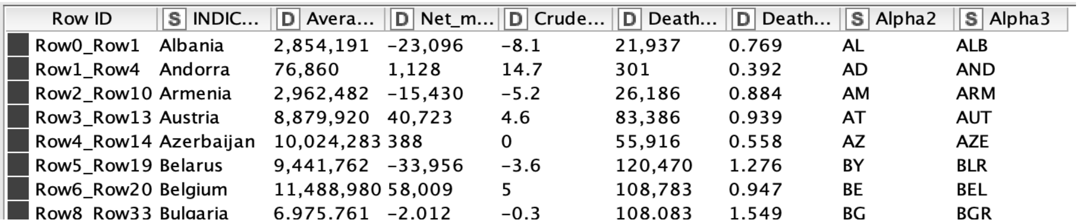
4.1.4. Missing values

After achieving the desired result from our Joiner node, we had to filter out the missing values. This was done using a Missing Value node which was configured to enter NA wherever it encountered a missing value as displayed in the output image.



Row ID	S INDIC...	D Avera...	D Net_m...	D Crude...	D Death...	D Death...	S Alpha2	S ▼ Alp...
Row19_?	European ...	513,858,...	1,680,681	3.3	5,257,654	1.023	NA	NA
Row25_?	Germany i...	83,092,962	308,928	3.7	939,520	1.131	NA	NA
Row31_?	Kosovo	1,788,891	-25,919	-14.5	9,430	0.527	NA	NA
Row37_?	Moldova	2,663,251	-41,232	-15.5	36,416	1.367	NA	NA
Row40_?	Netherlands	17,344,874	107,627	6.2	151,885	0.876	NA	NA
Row41_?	North Mac...	2,076,694	-276	-0.1	20,446	0.985	NA	NA

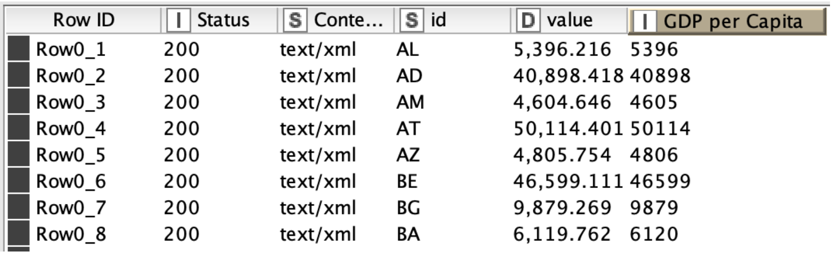
This step was followed by using a Rule Based Row Filter node that was configured to remove the observations that had NA values. Once this was done, a snippet of our final data table looked like this:



Row ID	S INDIC...	D Avera...	D Net_m...	D Crude...	D Death...	D Death...	S Alpha2	S Alpha3
Row0_Row1	Albania	2,854,191	-23,096	-8.1	21,937	0.769	AL	ALB
Row1_Row4	Andorra	76,860	1,128	14.7	301	0.392	AD	AND
Row2_Row10	Armenia	2,962,482	-15,430	-5.2	26,186	0.884	AM	ARM
Row3_Row13	Austria	8,879,920	40,723	4.6	83,386	0.939	AT	AUT
Row4_Row14	Azerbaijan	10,024,283	388	0	55,916	0.558	AZ	AZE
Row5_Row19	Belarus	9,441,762	-33,956	-3.6	120,470	1.276	BY	BLR
Row6_Row20	Belgium	11,488,980	58,009	5	108,783	0.947	BE	BEL
Row8_Row33	Bulgaria	6,975,761	-2,012	-0.3	108,083	1.549	BG	BGR

4.1.5. Data Cleaning processes on GDP/capita data table

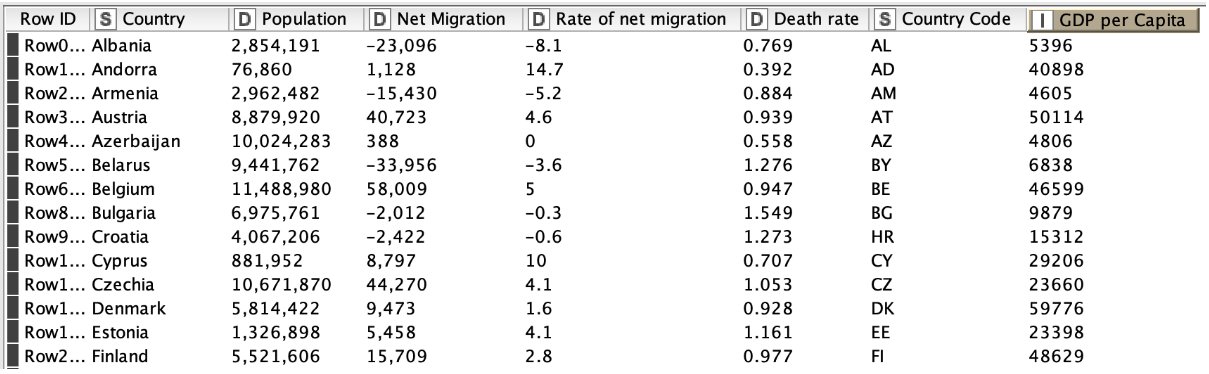
After importing GDP per capita data from World Bank and extracting into .xml format, we used a String to Number node to convert the values we received into numbers in order to perform our analysis on it, followed by a Math Formula node which was used to round the value for GDP/Cap and append it into the data table as a new column.



Row ID	Status	Conte...	id	value	GDP per Capita
Row0_1	200	text/xml	AL	5,396.216	5396
Row0_2	200	text/xml	AD	40,898.418	40898
Row0_3	200	text/xml	AM	4,604.646	4605
Row0_4	200	text/xml	AT	50,114.401	50114
Row0_5	200	text/xml	AZ	4,805.754	4806
Row0_6	200	text/xml	BE	46,599.111	46599
Row0_7	200	text/xml	BG	9,879.269	9879
Row0_8	200	text/xml	BA	6,119.762	6120

This cleaned data table received from the World Bank API was then joined with our previous output through a Joiner node.

This output was then filtered and renamed using a Column Filter node and Column Rename node. Taking a look at our final data table:



Row ID	Country	Population	Net Migration	Rate of net migration	Death rate	Country Code	GDP per Capita
Row0...	Albania	2,854,191	-23,096	-8.1	0.769	AL	5396
Row1...	Andorra	76,860	1,128	14.7	0.392	AD	40898
Row2...	Armenia	2,962,482	-15,430	-5.2	0.884	AM	4605
Row3...	Austria	8,879,920	40,723	4.6	0.939	AT	50114
Row4...	Azerbaijan	10,024,283	388	0	0.558	AZ	4806
Row5...	Belarus	9,441,762	-33,956	-3.6	1.276	BY	6838
Row6...	Belgium	11,488,980	58,009	5	0.947	BE	46599
Row8...	Bulgaria	6,975,761	-2,012	-0.3	1.549	BG	9879
Row9...	Croatia	4,067,206	-2,422	-0.6	1.273	HR	15312
Row1...	Cyprus	881,952	8,797	10	0.707	CY	29206
Row1...	Czechia	10,671,870	44,270	4.1	1.053	CZ	23660
Row1...	Denmark	5,814,422	9,473	1.6	0.928	DK	59776
Row1...	Estonia	1,326,898	5,458	4.1	1.161	EE	23398
Row2...	Finland	5,521,606	15,709	2.8	0.977	FI	48629

5. Data views

5.1. Visualization 1: GDP per Capita – statistical overview of the examined countries

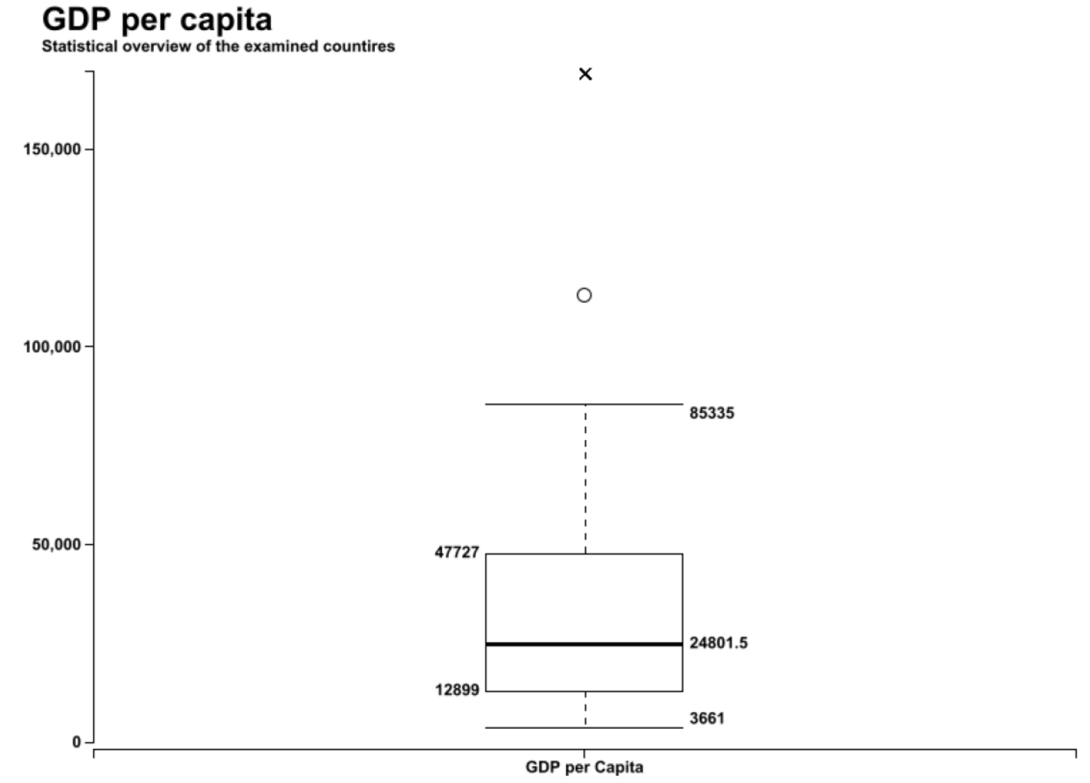
The purpose of this workflow is to show the overall statistics for GDP per Capita. This provides us with the summary statistics including minimum value, first quartile, median, upper quartile, and the maximum value.

Sorter:

the column GDP per Capita was first sorted in a descending order to create our box-plot visual.

5.1.1. Box Plot

With this visual, we can see the overall statistics of the GDP/cap column. This helps us understanding the values we obtained and building an intimation regarding our data.



5.2. Visualization 2: Rate of net migration view

The purpose of this workflow is to answer the question of, depending on the type of institution, what was the percentage of public or private expenditure by each country in the year 2011?

Top K Selector:

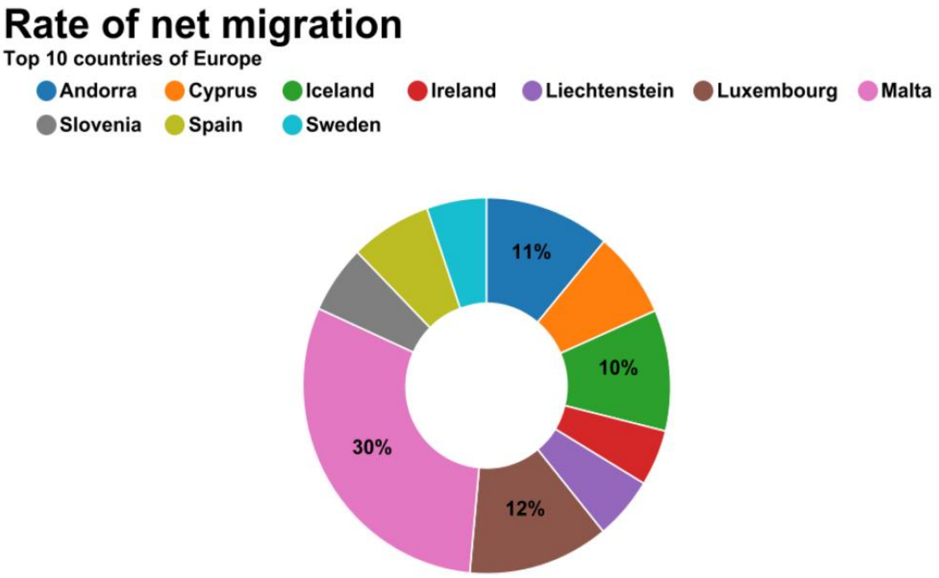
This node was used to filter out the top 10 countries on the basis of their net rate of migration.

Table View

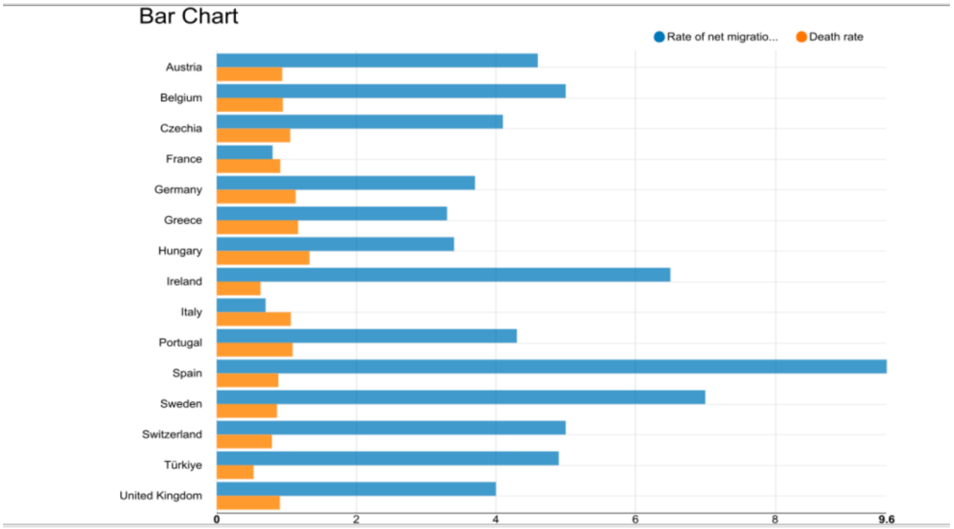
We wanted to analyze these countries having the highest net rate of migration therefore this node was used. It provided us with a table as an output which was assessed before sending it to the next node for visualization.

Pie/Donut Chart

This node creates a Pie/Donut Chart which gives a very good view of the countries that had the highest net rate of migration. It allows us to compare individual categories to the larger whole. The category column was configured as the country and our frequency column was the net rate of migration.

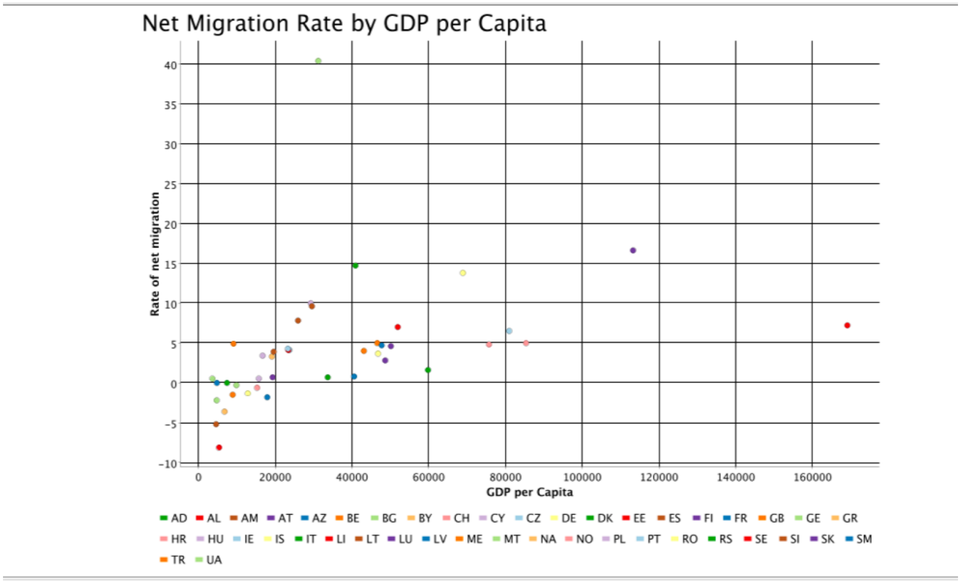


5.3. Visualization 3: Bar Chart



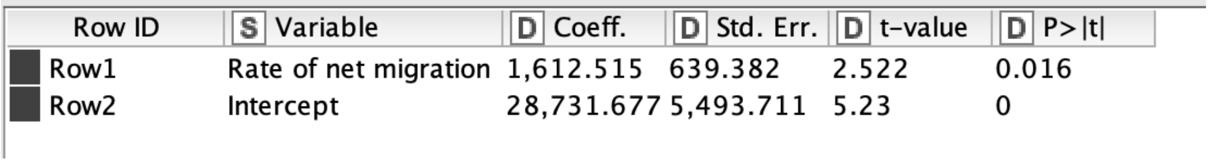
The bar chart generated shows the rate of net migration and the death rate. This is to compare if the countries with higher death rates show higher rate of net migration. A higher death rate can mainly be attributed to an ageing population and hence a reason for high net migration rates. Unique observations include Italy & France where the death rate exceeds the net migration rate. On the contrary, countries such as Sweden, Ireland, Spain & Türkiye have exceptionally high rates of migration compared to their death rates.

5.4. Visualization 4: Scatter Plot



The above scatter plot shows GDP per capita on the x-axis while the y-axis shows Rate of net migration. This is to spot a relation between GDP per capita and rate of net migration. Most countries lie in the first quadrant of the plot showing a relatively week relationship between the rate of net migration & GDP per capita.

5.4.1. Linear Regression



Row ID	S Variable	D Coeff.	D Std. Err.	D t–value	D P>|t|
Row1	Rate of net migration	1,612.515	639.382	2.522	0.016
Row2	Intercept	28,731.677	5,493.711	5.23	0

Lastly, a linear regression is performed on GDP per capita by Rate of Net Migration. The regression table shows the coefficients, standard error & t- value between rate of net migration & GDP per capita.