CITS5504 Data Warehousing

Project Report

Varun Jain (21963986)

Semester 1 2021

Contents

Design Implementation

1.1 Concept Hierarchies & Starnet Model

Before developing a starnet model, we first had to analyse the COVID-19 dataset and build a concept hierarchy for each dimension. In Figure 2, each line in the starnet model represents a concept hierarchy of total order, with a root node ALL at the top. In Figure 1, the concept hierarchy for location is categorised to four distinct levels, "ALL", "Region", "Continent" and "Country", to which later it will be mapped onto the starnet model.

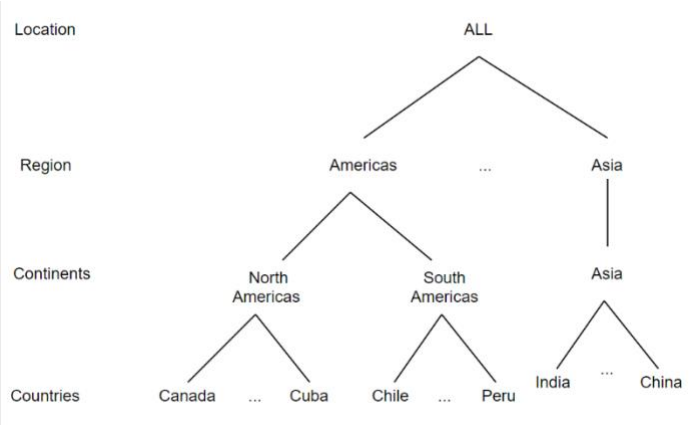


Figure 1. Concept Hierarchy

The Starnet model in Figure 2, consists of four radial lines, where each line represents a different level of granularities available in the model, meaning the degree of generalisation or specialisation of information changes based on the abstraction level. The granularities change depending on the OLAP operations applied to the model. For instance, in the location hierarchy in the Starnet model, the users can use rollup to generalise data, from *continent* to *region*, or drill down to further specialise and get access to lower-level values, from *continent* to *countries*.

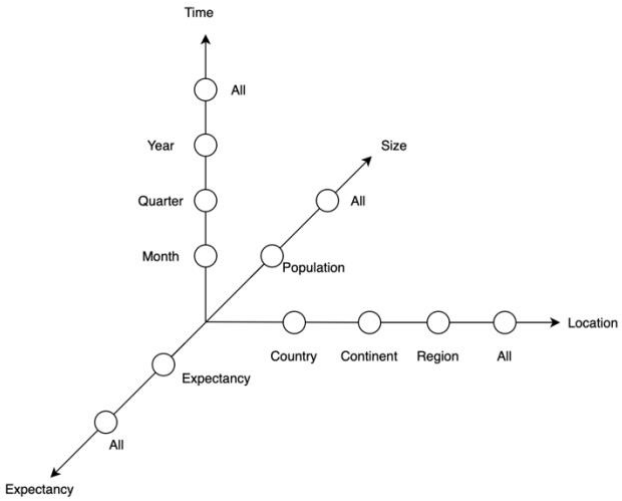


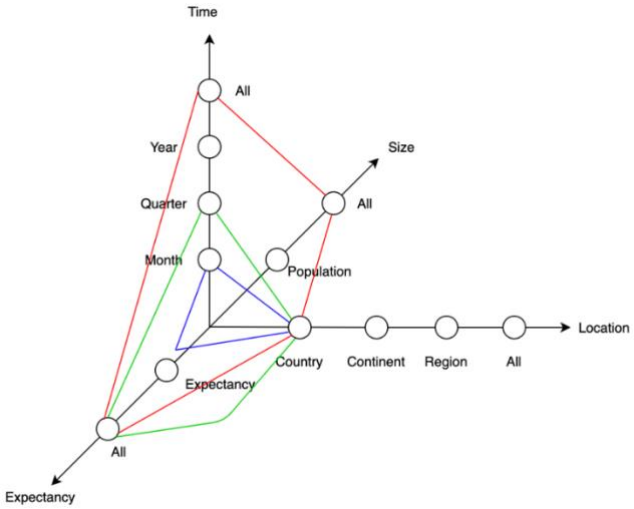
Figure 2: Starnet Model

1.2 Business Queries and Starnet Footprints

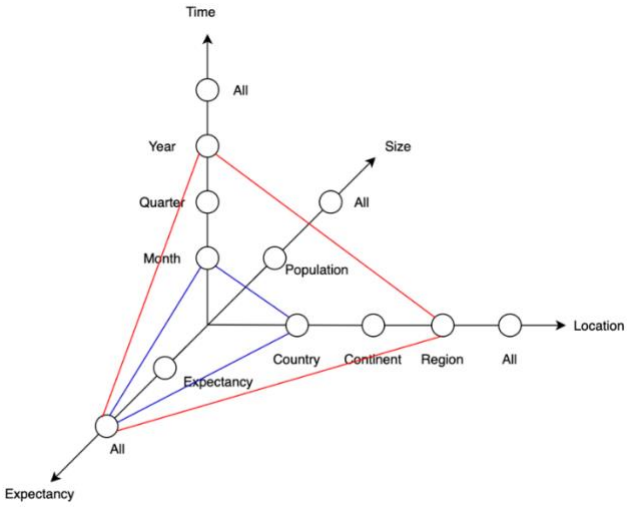
The starnet model was designed and remodelled multiple times to answer the following business queries provided by the UWA Team of Data Warehousing. Each business query may be broken down into multiple inquiries and with the appropriate StarNet footprint model drawn below.

The Business Queries:

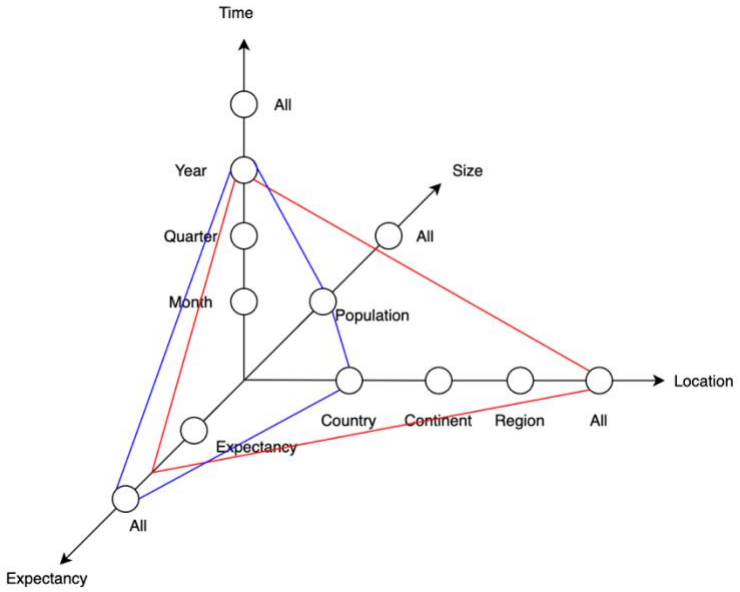
1. What is the total number of confirmed cases in Australia? (shown in red) What is the number of cases in each quarter of 2020 in Australia? (shown in green) What is the number of confirmed cases in each month of 2020 in Australia? (shown in blue).



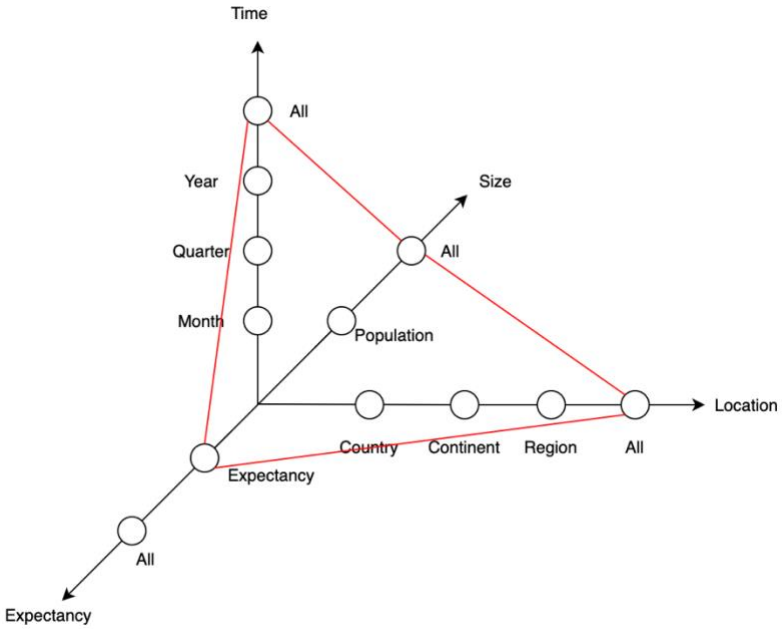
2. In September 2020, how many recovered cases are there in the region of Americas? (shown in red). How many recovered cases in the United States, Canada, and Mexico, respectively, in sept 2020? (shown in blue).



3. What is the total number of covid deaths worldwide in 2020? (shown in red) What is the total number of covid deaths in large countries, medium countries, and small countries, respectively in 2020? (shown in blue).



4. Do countries with a life expectancy greater than 75 have a higher recovery rate?



1.3 Star Schema

The database design, in Figure 3, is a model based on the Star Schema. The schema contains four-dimension tables centralised around the fact table containing the following measures: *confirmed cases*, *death cases* and *recovery cases*, and a key to each of the following dimension tables: *country*, *time*, *size*, and *expectancy*.

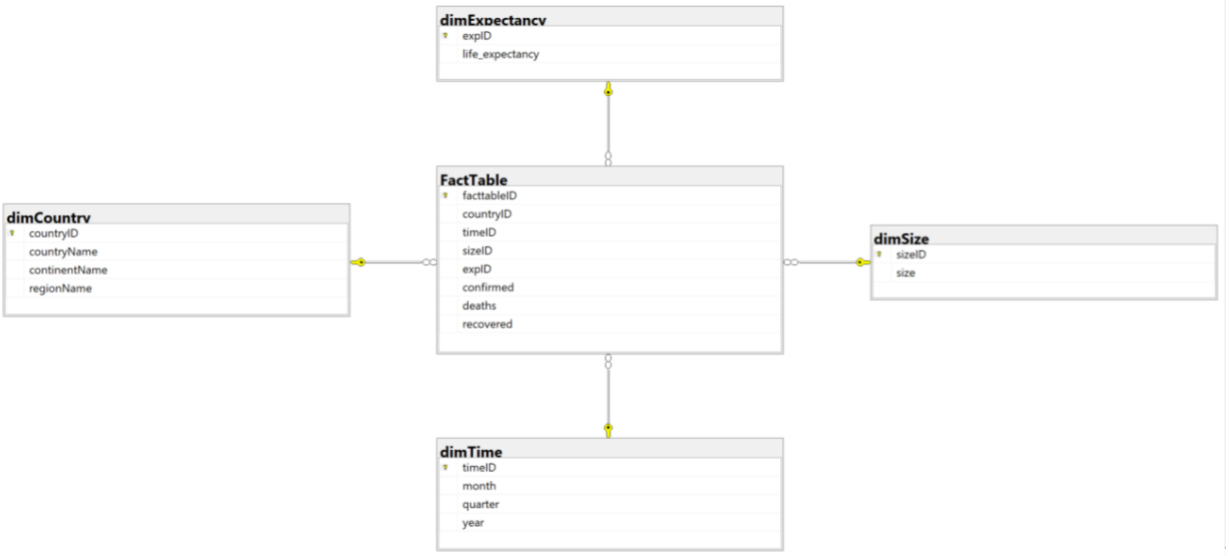


Figure 3. Star Schema

1.4 Extract, Transform, Load (ETL)

The inconsistency and the variability in and between the COVID-19 datasets were quite significant. For the ETL process, I used both python, excel and sql to extract, transform, and load the appropriate data to the csv files, then to the .sql scripts to create and populate the database in the SSMS server.

The initial step of the ETL process was to analysis the data and extract all the relevant information we required to answer the business queries. From our observations, the .csv case files for *confirmed*, *deaths* and *recovery* had to be remodelled and structured in an appropriate manner to be able to extract the required information.

```
def read_and_melt(filename, covid_type):
    covid_cases = pd.read_csv(filename)
    covid_cases = pd.melt(covid_cases, id_vars=covid_cases.columns[:4],
                          value_vars = covid_cases.columns[4:],
                          var_name = 'date',
                          value_name = covid_type)
    return covid_cases

confirmed = read_and_melt('time_series_covid19_confirmed_global.csv','confirmed')
deaths = read_and_melt('time_series_covid19_deaths_global.csv','deaths')
recovered = read_and_melt('time_series_covid19_recovered_global.csv','recovered')
```

Figure 4. Melt Function

Each .csv file is read through the pandas() library function read_csv() then processed further to the melt function. Refer to Figure 4, the pandas function melt (…) is used to reshapes the design of the .csv files from a wide format to a long data frame. The COVID-19 case files are in time series format, where each date was allocated into separate columns recording the number of cases across the spreadsheet. The melt function skips over the first four columns in the .csv files and extracts all the *date columns*. The date columns are processed, and unpivoted to the row axis leaving two columns, the date and confirmed cases.

There were noticeable discrepancies with Canada in *…recovered_global.csv* compared with *death_global.csv* and *confirmed_global.csv* cases files. In the death and confirmed file, Canada's case records were partitioned into separate states while the recovery dataset recorded the number of individuals who recovered from COVID-19 as a whole, a total of all the states combined.

```
def covid_template(confirmed, deaths, recovered):
    #extracts canada's covid cases from each .csv file, skips all other countries and returns the total
    #number of covid cases for each date.
    confirmed_canada = summarise_country_by_date(confirmed, 'Country/Region', 'Canada', 'date', 'confirmed')
    deaths_canada = summarise_country_by_date(deaths, 'Country/Region', 'Canada', 'date', 'deaths')

    #Returns all the columns expect the confirmed cases column - this is used as a template
    reovered_canada = extract_tables(recovered, 'Country/Region', 'Canada')

    #The template created in merges with the confirmed cases, and deaths cases.
    canada = merge_columns(reovered_canada, confirmed_canada, 'inner', 'date', True)
    deaths_canada = merge_columns(reovered_canada, deaths_canada, 'inner', 'date', True)

    #remove the original data from the dataset, and append the new Canada's data
    confirmed = confirmed[confirmed['Country/Region'] != 'Canada'].append(canada)
    deaths = deaths[deaths['Country/Region'] != 'Canada'].append(deaths_canada)

    #merge together all three cases types into one dataframe.
    on = ['Province/State','Country/Region','Lat','Long','date']
    data = merge_cases(confirmed, deaths, 'left', on)
    data = merge_cases(data, recovered, 'left', on)

    return data
```

Figure 5. Covid Template

To fix the issue, the following code, shown above, was implemented to normalise the data to a standard COVID-19 template by removing any sort of discrepancy within the data. It basically extracts and transforms Canada's data for confirmed and death cases, from individual counts of each state to a summation of all the states for the particular dates. I added an additional feature to append the confirmed, recovered, and deaths covid case into the data frame. The functions implemented within the covid_template(…) can be found in the python script etl.py.

From the COVID-19 template, the following ships, Dimond Princess, MS Zaandam and Grand Princess have been removed from the data frame. The total impact on the total number of cases worldwide was very negligible but also there was a lot of confusion on how to make use of the data. There was a degree of variability to the way data was stored in the data frame. For instance, Diamond Princess, was stated as a Country but also a State, which was mapped to Canada. There was no definite country assigned to individual boats, so I was not sure how the ships were integrated in the dataset.

The discrepancies in the naming conventions and the list of countries, was astonishingly more than I would have imagined. The three files, government_measures_dataset.xlsx, owid-covid-data.csv and the covid template, had its own list of countries, and spelling. The aim was to transform the data where the list of countries in all three data frames matched. Initially, I compared the government measures dataset with the owid-covid-data and subtracted away all the countries that were not present in the government measures and replaced any acronyms or misspelt countries to its proper naming conventions. The same process was applied with the new formed country list with covid template country list. The reasoning behind this was to ensure that for each country, the region, life expectancy and population was uniformly matched.

For the *time* dimension table, we separated the datetime into four separate columns using the following code, shown below. Four new columns are created: day, month, quarter and year. So, using the pandas library function pd.to_datetime(…), it converts the string "1/1/2020" into datetime. Then we use the following attributes *.dt.day*, *.dt.month_name()*, *dt.quarter* and dt.year to return the day, month name, the quarter and the year of the datetime.

```
cases_combined['date']= pd.to_datetime(cases_combined['date'])
cases_combined['day'] = cases_combined['date'].dt.day
cases_combined['month'] = cases_combined['date'].dt.month_name().str.slice(stop = 3)
cases_combined['quarter'] = cases_combined['date'].dt.quarter
cases_combined['quarter'] = 'Q' + cases_combined['quarter'].astype(str)
cases_combined['year'] = cases_combined['date'].dt.year
cases_combined[cases_combined['month'] == 'Feb']
```

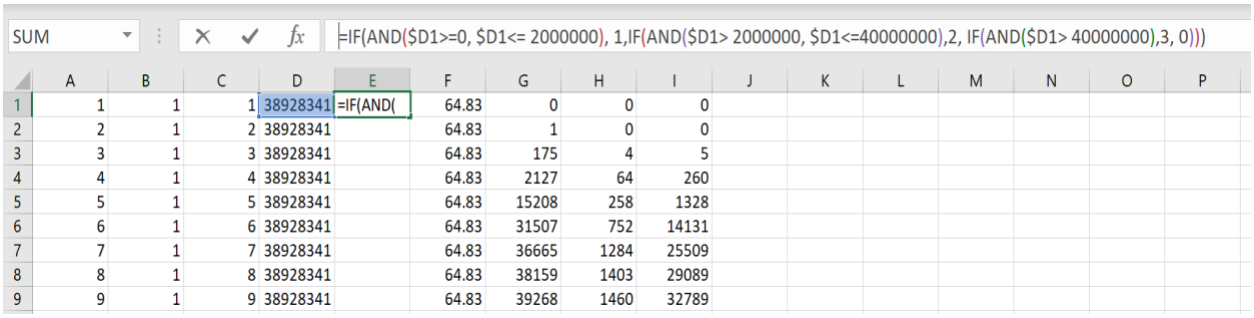
Based on the business queries questions, it requires to compute results for month, quarter, and year. For simplicity, the day column is dropped as it will not help us in answering any of the four business queries. Further, we drop the duplicate rows as we only require unique rows to store in the dimension table. Therefore, 15 unique rows are creating where the month stores the month name, quarter stores the quarter period of the month, and the year.

```
#drop all the columns except month, quarter and year, and return all the unique values for the group.
date_time=cases_combined.drop(cases_combined.columns[0:9], axis = 1).drop_duplicates()
date_time_dataframe = pd.DataFrame(date_time)
#Add a unique id for each row
date_time_dataframe.insert(0, 'Id', range(1, 1 + len(date_time_dataframe)))
```

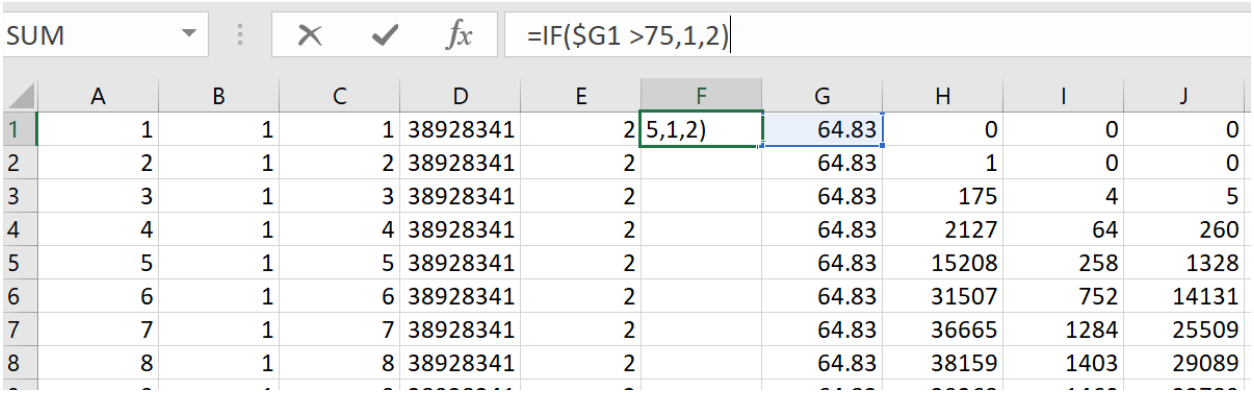
Fact table consists of the measurements and the foreign keys for the relevant dimension table. The fact table measures the total number of COVID-19 instances for death, recovery and confirmed. I have transformed my database to account to such that it will only store the output the rows for the last day of each month for each country. That way the total number of cases are stored cumulatively, meaning that having the operations of each day does not provide us with any additional information, which reduces the size of the data frame significantly. This will simplify the data frame to get exactly what we need to answer the business query. Then, merge the time dimension, country life expectancy and population into the data frame and remove any unwanted columns and rearrange the data frame into an appropriate format. Finally, using Pandas .to_csv() to output the fact table to a .csv file. The end result can be seen in Figure 10.

	countryID	timeID	population	life_expectancy	confirmed	deaths	recovered
0	1	1	38928341.0	64.83	0	0	0.0
1	1	3	38928341.0	64.83	175	4	5.0
2	1	5	38928341.0	64.83	15208	258	1328.0
3	1	7	38928341.0	64.83	36665	1284	25509.0
4	1	8	38928341.0	64.83	38159	1403	29089.0
...	...	...	...	...	...	...	...
3509	182	14	14862927.0	61.49	36089	1463	32666.0
3510	182	4	14862927.0	61.49	40	4	5.0
3511	182	6	14862927.0	61.49	591	7	162.0
3512	182	9	14862927.0	61.49	7838	228	6303.0
3513	182	11	14862927.0	61.49	9950	276	8482.0

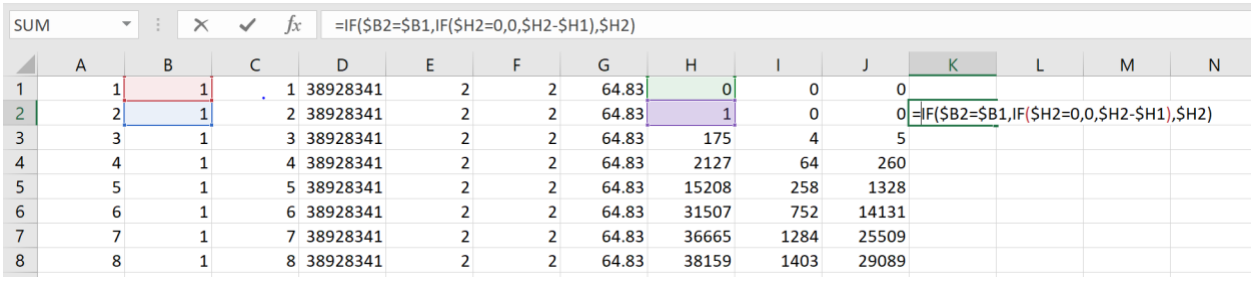
Further data processing and transformation will take place using excel. First create a new column to store the values of your computation. In Figure 11, the IF statement loops through the population row, making comparisons, and checking if the IF statement is meeting any of the three criteria. Depending on the IF condition that is satisfied, the IF statement will output either 1, 2 or 3 on to empty column. 1 indicates that the population for that specific country is less than 2 million, 2 indicates that the population is somewhere between 2 million and 40 million, and 3 meaning the population is greater than 40 million. The value 1, 2, and 3 are linked to the dimSize, with the following values, “small,” “medium” and “large”.



Similar computation is computed for the life_expectancy but instead, the IF statement allocates either 1 or 0, where 1 indicates that the life expectancy average is greater than 75, otherwise, 0.



To convert the accumulate data into unaccumulated date, we compute the IF statement shown in the figure below. The IF statement basically compares the countryID for the second and third cell, and if they match, subtract the covid case instances from the third row with the second.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	1	1	1	38928341	2	2	64.83	0	0	0				
2	2	1	2	38928341	2	2	64.83	1	0	0	=IF($B2=$B1,IF($H2=0,0,$H2-$H1),$H2)			
3	3	1	3	38928341	2	2	64.83	175	4	5				
4	4	1	4	38928341	2	2	64.83	2127	64	260				
5	5	1	5	38928341	2	2	64.83	15208	258	1328				
6	6	1	6	38928341	2	2	64.83	31507	752	14131				
7	7	1	7	38928341	2	2	64.83	36665	1284	25509				
8	8	1	8	38928341	2	2	64.83	38159	1403	29089				

Repeat the Step for the remaining two covid measure columns. Finally remove population, life expectancy and the cumulative covid cases as there is now no need to have them in our Fact Table.

1.5 Implementation

1.5.1 Load Database

The insert.sql script can be used to create the covid19 database on the sql server. The script will create the fact table, with the dimensions table and it's appropriate referencing between the foreign and primary key. When the following .sql code is executed, it will deleted any existing database prior to this execution and create a new databased called warehouse.

```
PRINT '';
PRINT '*** Dropping Database';
GO

IF EXISTS (SELECT [name] FROM [master].[sys].[databases] WHERE [name] = N'warehouse')
DROP DATABASE warehouse;
GO

PRINT '';
PRINT '*** Creating Database';
GO
```

To create tables, we utilise the sql command CREATE TABLE. Columns are defined within the function to declare the attributes, its datatype and whether it can accept null values and each column ensuring that the data is consistent. As you can see in the following block of code, a dimension table is created called dimCountry. The dimension table stores a primary key countryID, to unique identify each country in the table , and the following fields as seen below with a NOT NULL constraint.

```
PRINT '';
PRINT '*** Creating Table dimTime';
GO

CREATE TABLE dimTime (
        timeID INT PRIMARY KEY,
        month VARCHAR(50) NOT NULL,
        quarter VARCHAR(50) NOT NULL,
        year INT NOT NULL
)
GO
```

Since the design of the database follows a star schema, the foreign key constraints must call link up to the fact table. Each dimension table must have its own primary key, to unique identify the values within the dimension table. The foreign keys will ensure referential integrity that the relationship between the fact and dimension key is maintained.

```
AlTER TABLE FactTable ADD CONSTRAINT
FK_countryID FOREIGN KEY (countryID)REFERENCES dimCountry(countryID);
AlTER TABLE FactTable ADD CONSTRAINT
FK_timeID FOREIGN KEY (timeID)REFERENCES dimTime(timeID);
AlTER TABLE FactTable ADD CONSTRAINT
FK_expID FOREIGN KEY (sizeID)REFERENCES dimSize(sizeID);
AlTER TABLE FactTable ADD CONSTRAINT
FK_sizeID FOREIGN KEY (expID)REFERENCES dimExpectancy(expID);
```

1.5.2 Populate the database

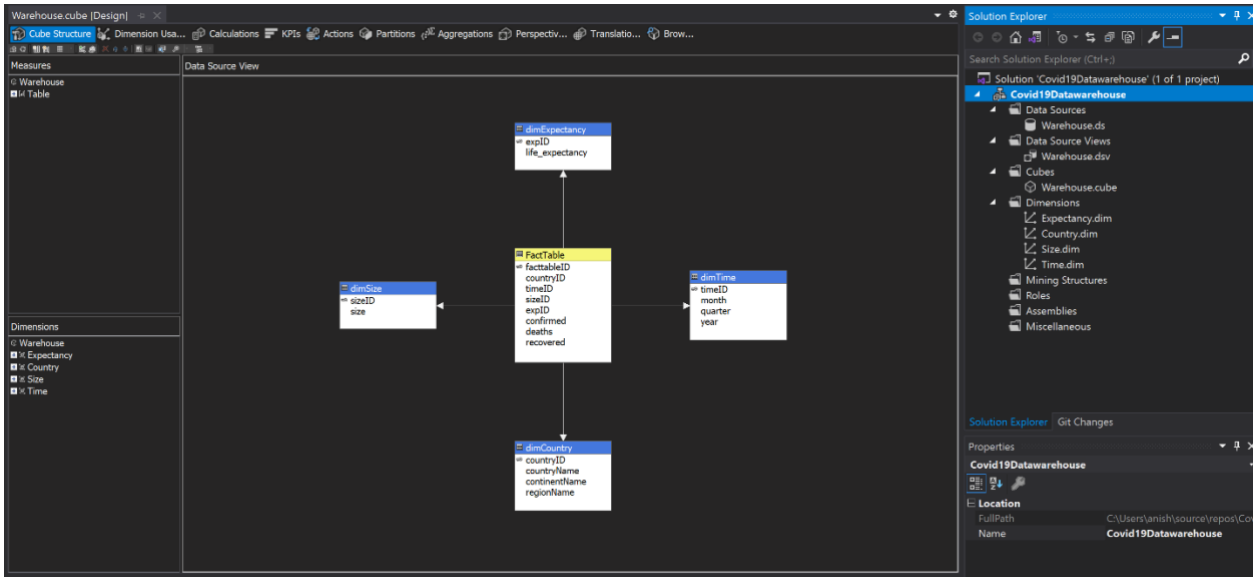
Populate the table exporting the processed .csv files to the sql dimensions and facts table using the populate.sql script. The code shown below is referenced from lab04. BULK INSERT takes processed data from the csv files and inserts them into the appropriate tables. The path of the directory where the .csv files are stored, are assigned to SqlSamplesSourceDataPath. Enable SQLCMD before running the script in SSMS.

```
:setvar SqlSamplesSourceDataPath "C:\Users\anish\OneDrive - The University of Western Australia\Desktop\dw\"
:setvar DatabaseName "warehouse"
BULK INSERT [dbo].[dimCountry]
FROM '$(SqlSamplesSourceDataPath)dimCountry.csv'
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='0x0a'
    --KEEPIDENTITY,

);
```

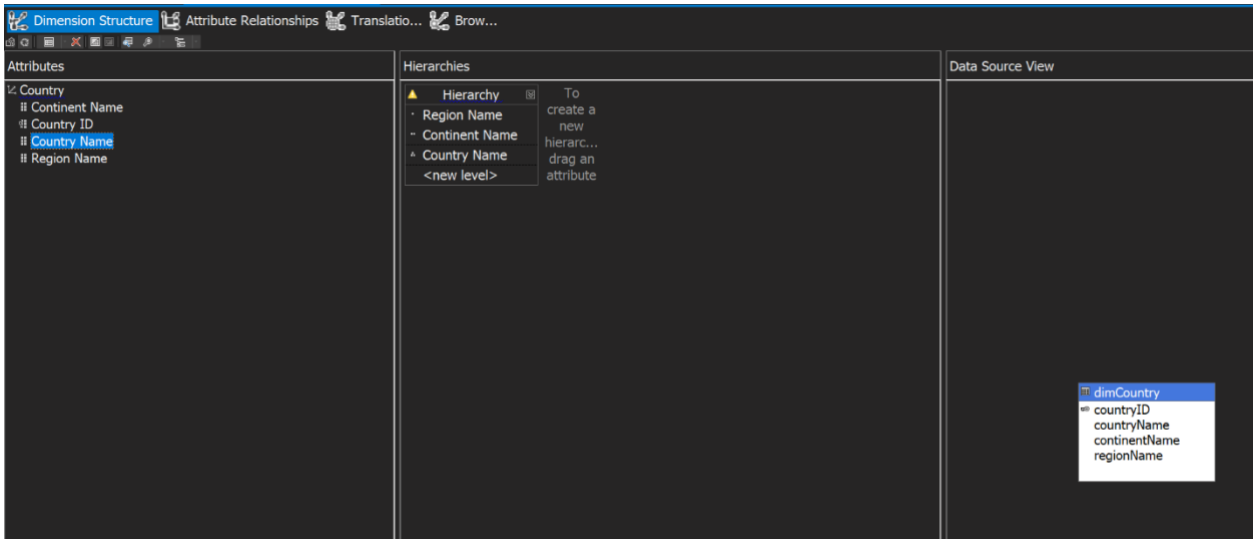
2.0 Data Cube and PowerBI

2.1 Data Cube Visualisation



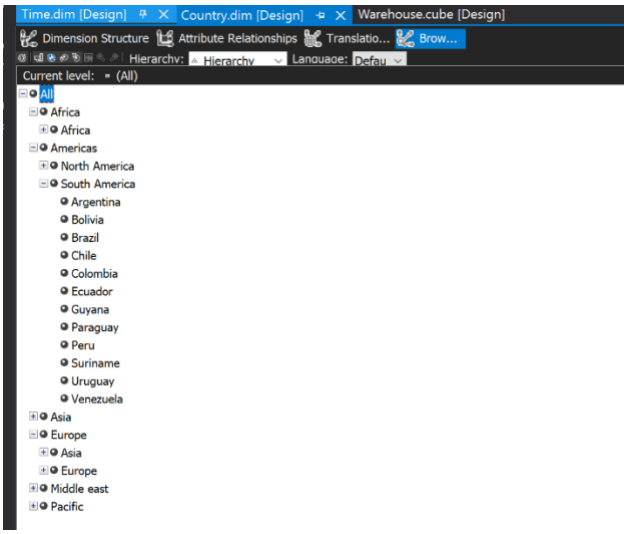
Referencing to the figure above, I have generated a data cube representation using visual studio for the database schema. The implementation of a data cube allows more versatile operations and outranks a normal sql database due to its restricted nature.

2.2 Data Cube Hierarchy



It creates meaningful concept hierarchies which to develop a relationship. A hierarchical relationship is created between region, continent, and country. We can use OLAP operations such as drill down to receive more detailed information each continent within a region or a which country are associated to what regions.

After you successfully link the countries hierarchy, the following figure shows the relationship between the region, continents, and countries. We perform drill up and drill down operations to see the relationships between the attributes.

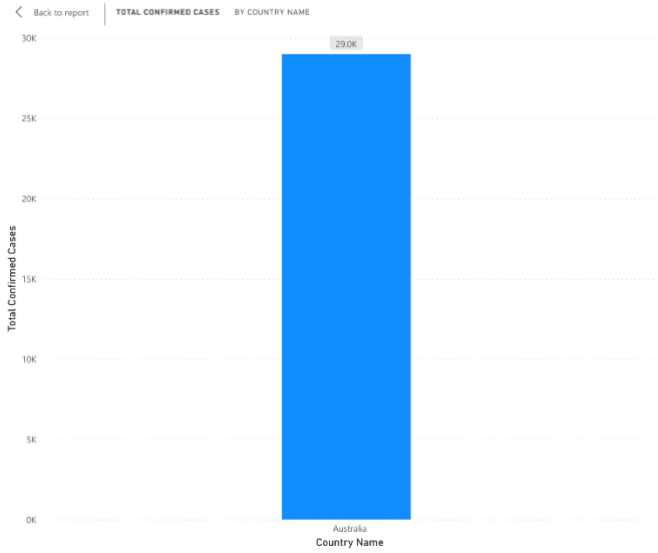


2.3 Business Queries

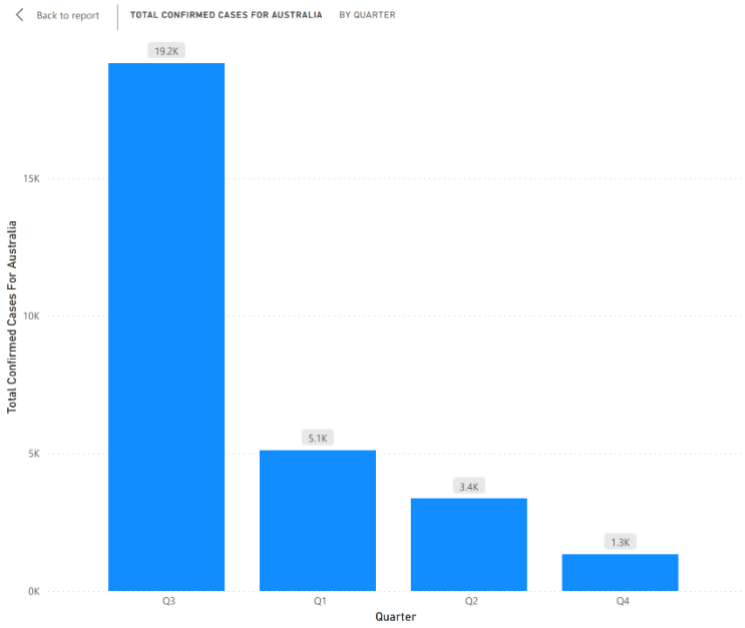
For the following Business Queries, we used OLAP operations such as slice, drill down and drill up to compute the following visualisations to answer the business queries.

2.3.1 Business Query 1

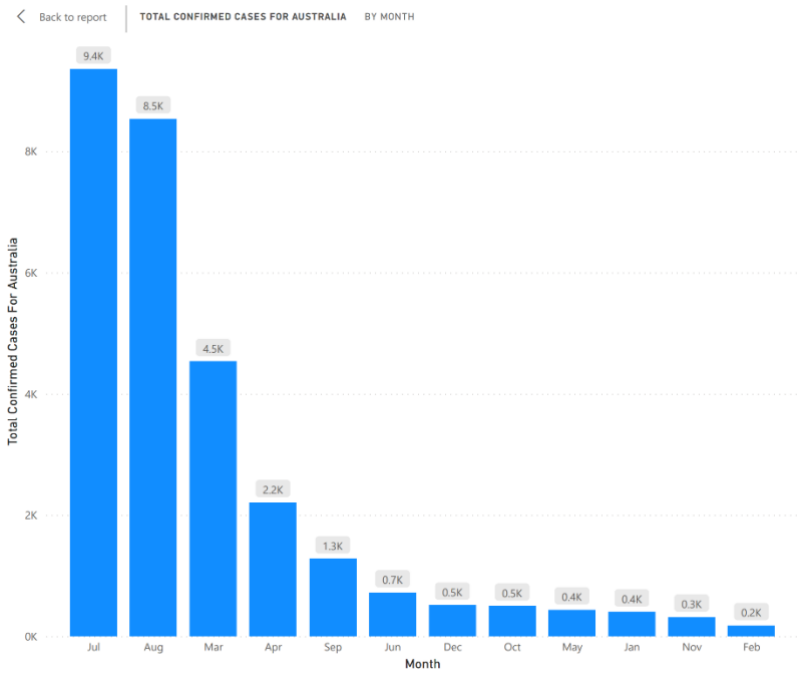
1. What is the total number of confirmed cases in Australia?



2. What is the number of cases in each quarter of 2020 in Australia?

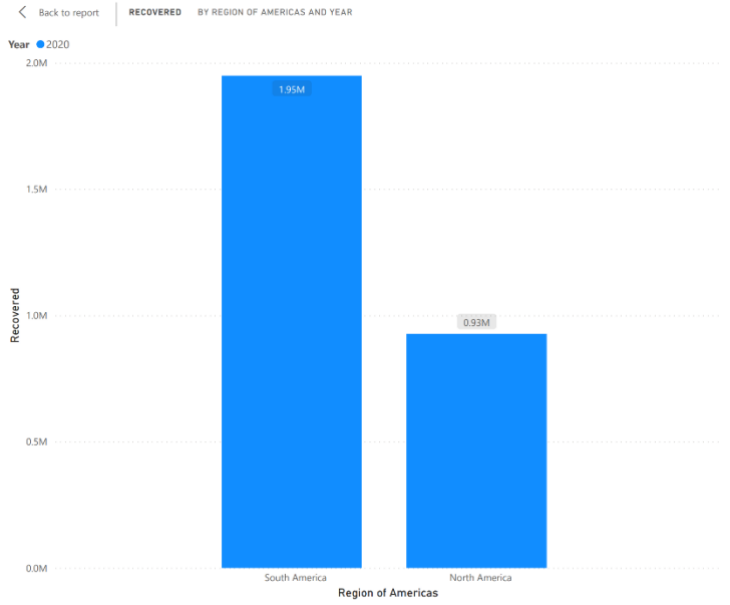


3. What is the number of confirmed cases in each month of 2020 in Australia?

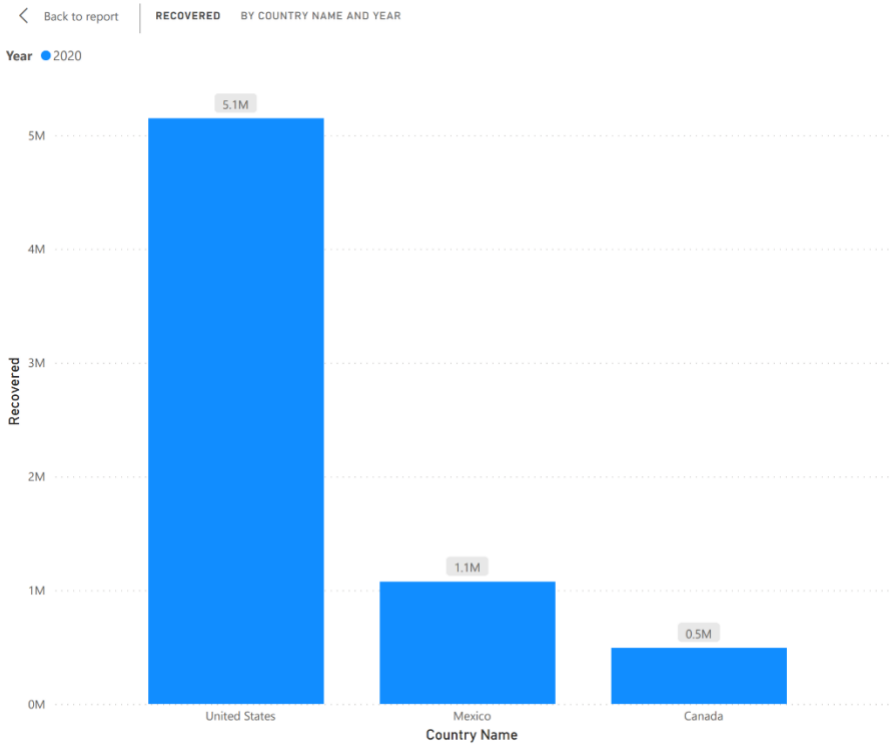


2.3.2 Business Query 2

1. In Sept 2020, how many recovered cases are there in the region of Americas?

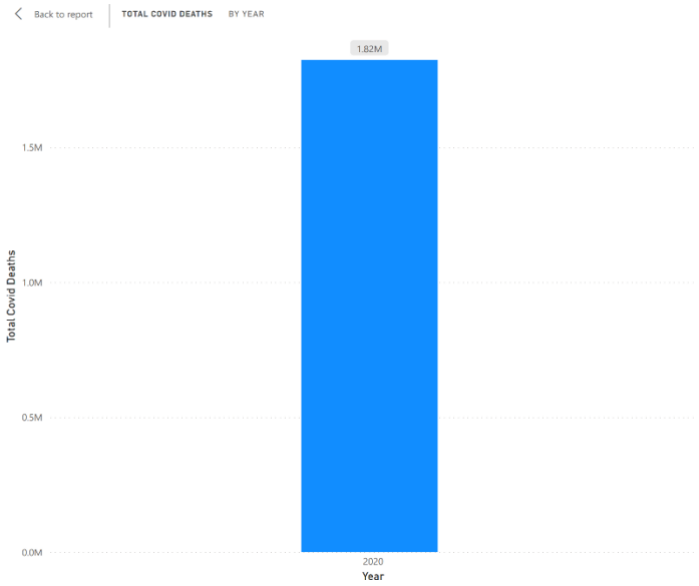


2. How many recovered cases in the United States, Canada, and Mexico, respectively, in sept 2020?

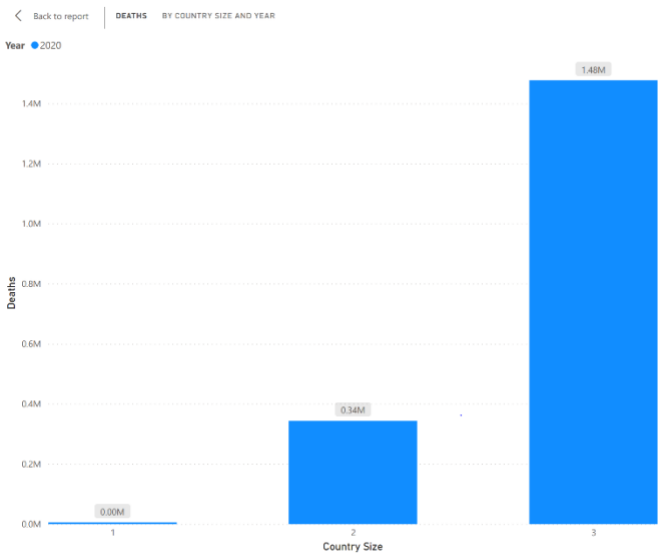


2.3.3 Business Query 3

1. What is the total number of covid deaths worldwide in 2020?



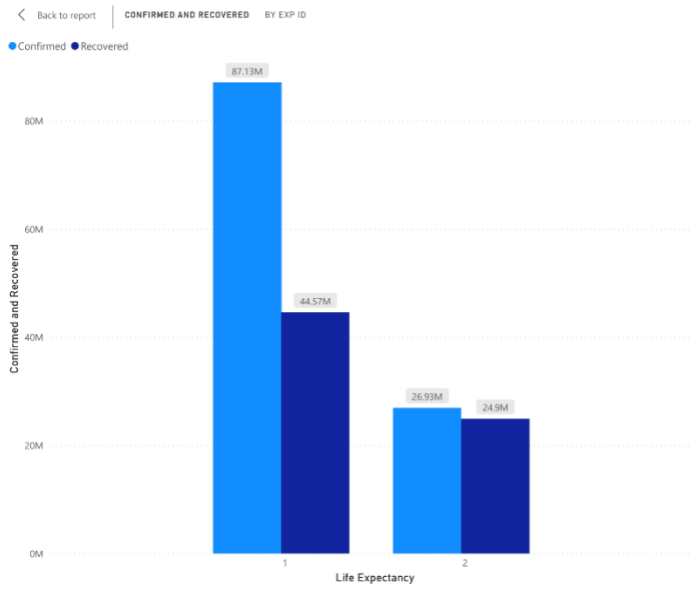
2. What is the total number of covid deaths in large countries, medium countries, and small countries, respectively in 2020?



Note: 1 indicates that country size is small. 2 indicates that country size is medium and 3 indicates that country size is large.

2.3.4 Business Query 4

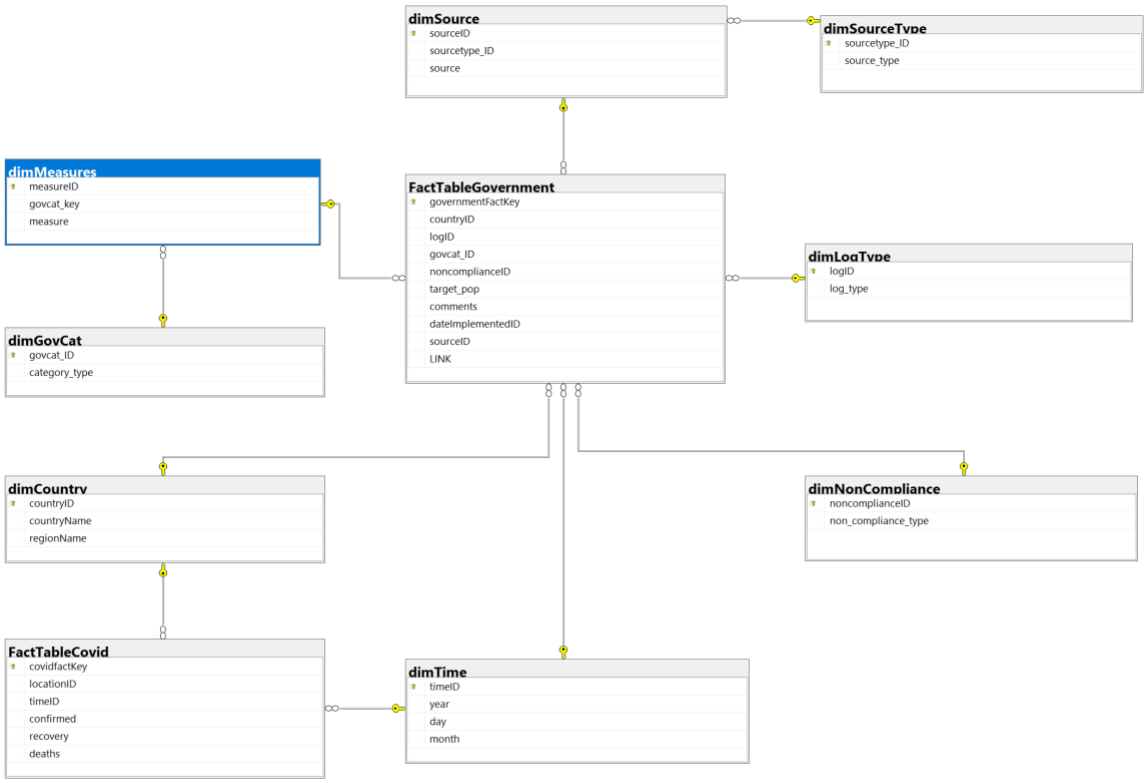
1. Do countries with a life expectancy greater than 75 have a higher recovery rate?



Note: 1 indicates life expectancy of less than 75, and 2 indicates life expectancy of greater than 75.

Countries with a life expectancy do have a higher recovery rate. This is noticeable in the visualisation as the recovery and confirmed cases are relatively very close to each other but for countries with a life expectancy lower than 75, the margin between the two types of cases are substantial.

3.0 Galaxy Schema



Which measure had the highest number of reports? And within that measure, what percentage of the reports came from the Governent?