

CITS3401 Data Warehousing  
Project 1 Report  
Hari Vignesh Amirthalingam (22874425)  
Semester 1 2021

# 1. Design and Implementation

## 1.1 Concept Hierarchies & StarNet Model

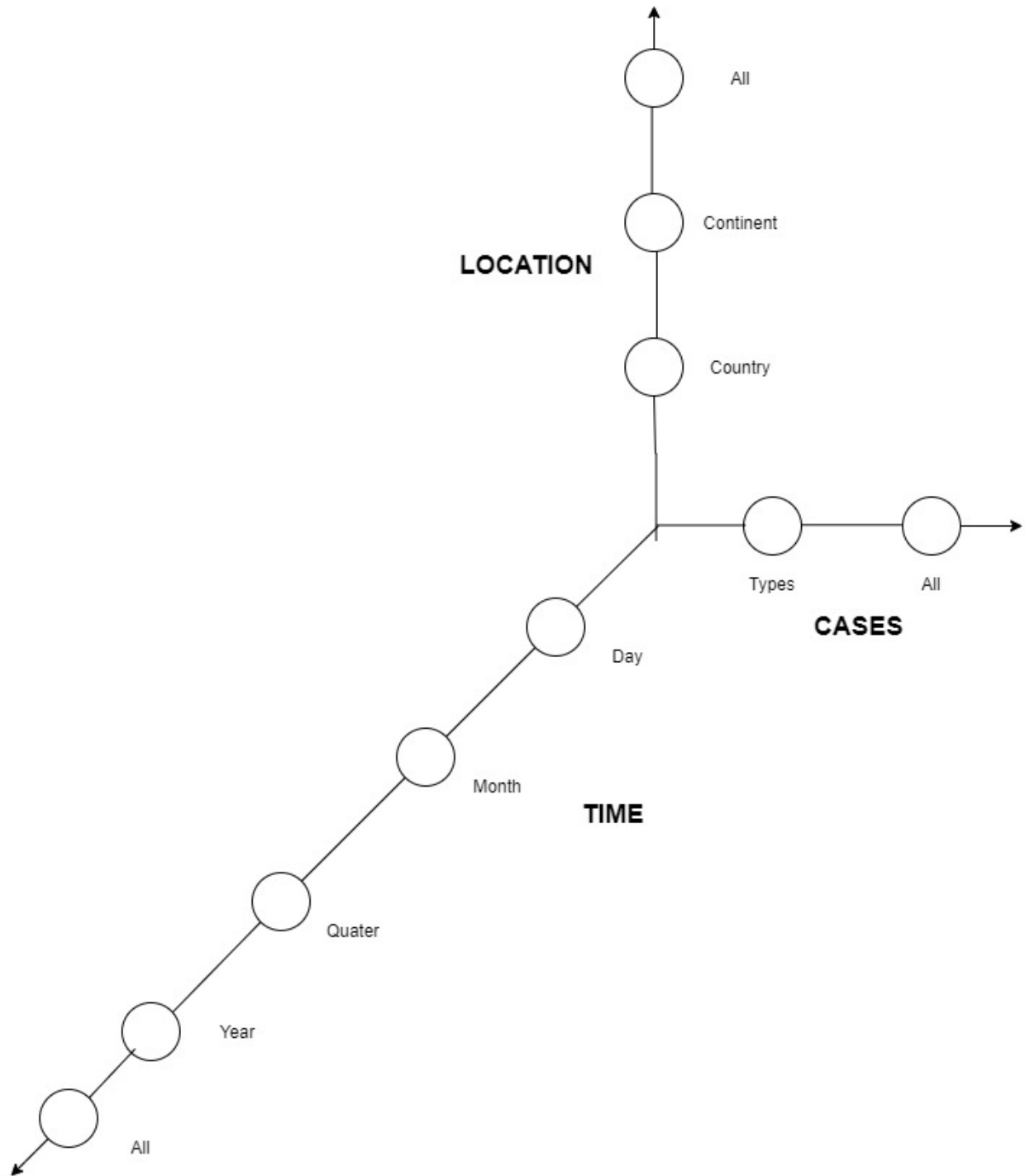


Figure 1

# 1.2 Business Queries & StarNet Footprints

The business Queries listed below are could always be answered by the StarNet Footprints regardless of their complexity.

The Business queries:

1. Queries:
  - a. What is the total number of confirmed cases in Australia in 2020?
  - b. What is the number of confirmed cases in each quarter of 2020 in Australia?
  - c. What is the number of confirmed cases in each month of 2020 in Australia?
2. Queries:
  - a. In Sept 2020, how many recovered cases are there in the region of the Americas?
  - b. How many recovered cases in the United States, Canada and Mexico, respectively, in Sep 2020?
3. Queries:
  - a. What is the total number of covid deaths worldwide in 2020?
  - b. What is the total number of covid deaths in large countries, medium countries and small countries, respectively, in 2020?
4. Do countries with a life expectancy greater than 75 have a higher recovery rate?

The StarNet footprints for their corresponding Queries are below:

1.a

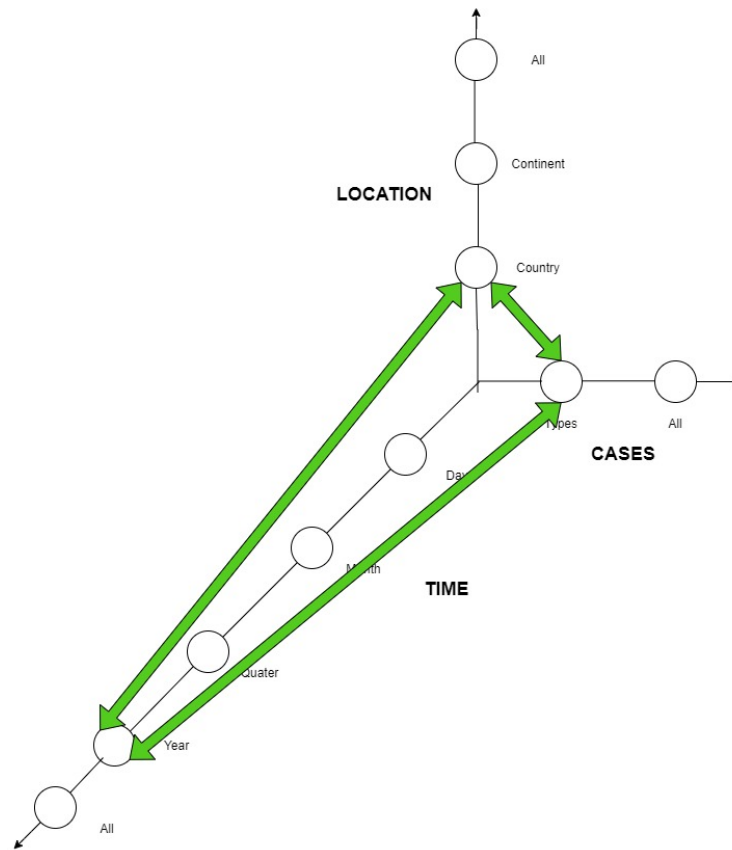


Figure 2

1.b

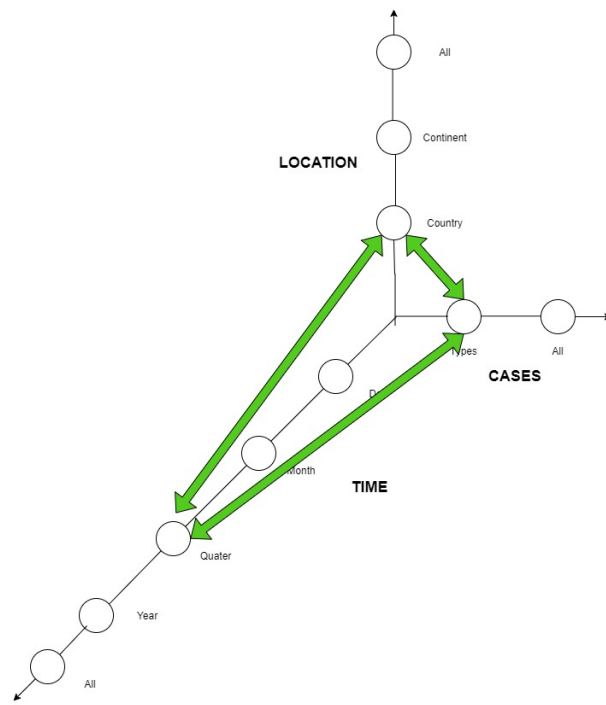


Figure 3

1.c

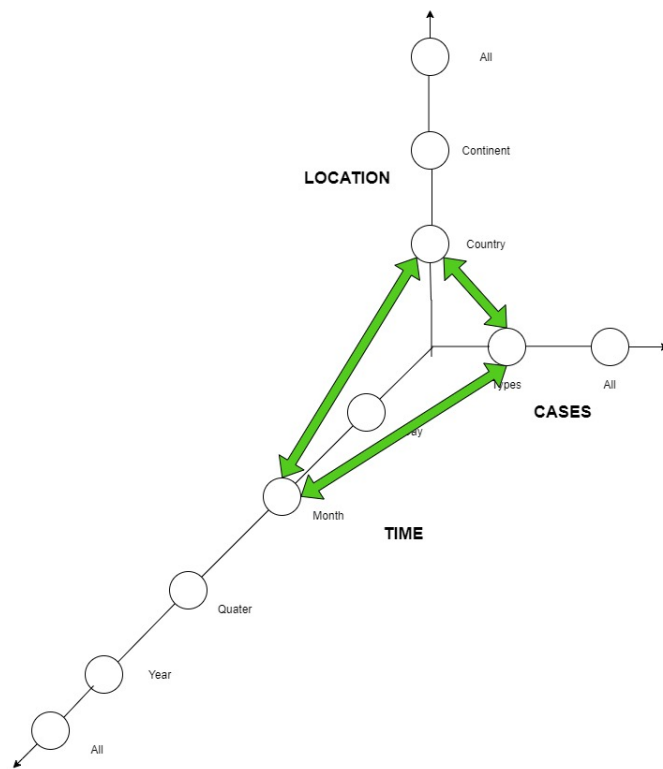


Figure 4

2.a

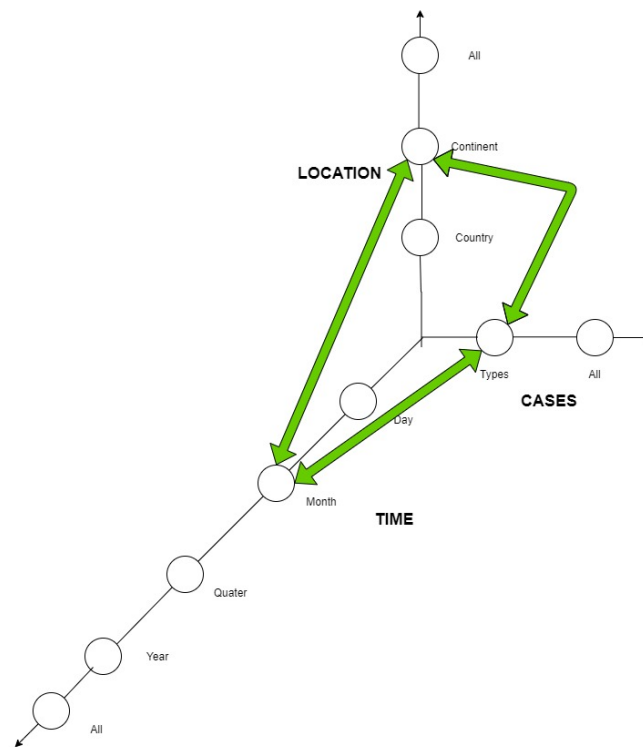


Figure 5

2.b

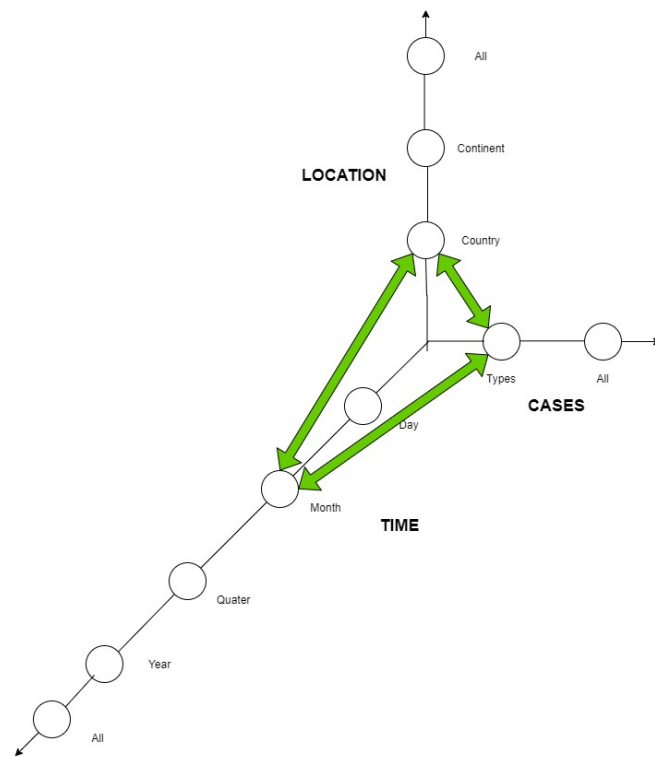


Figure 6

3.a

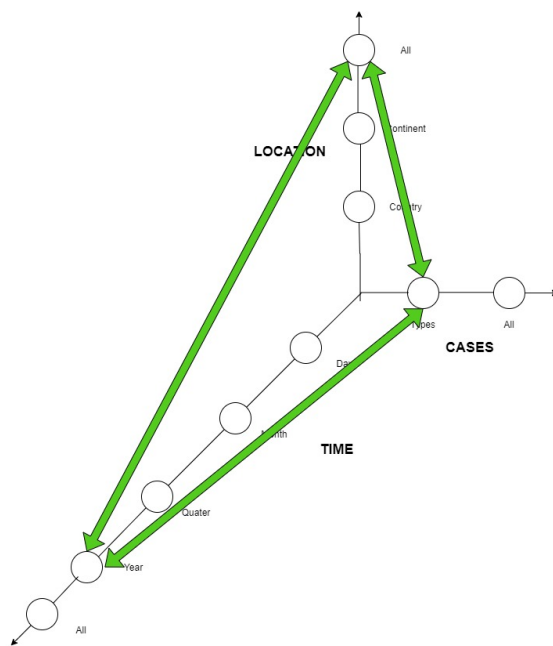


Figure 7

3.b

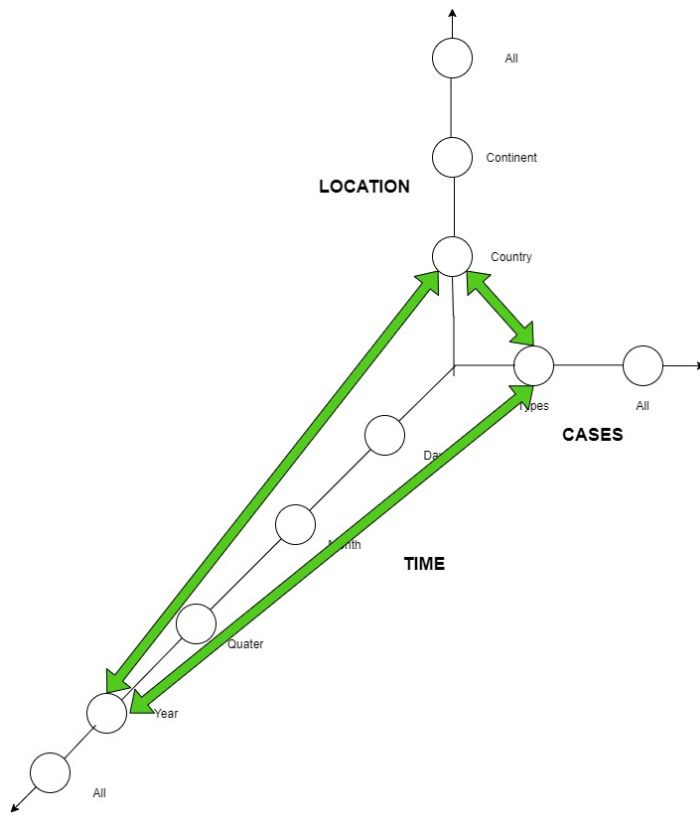


Figure 8



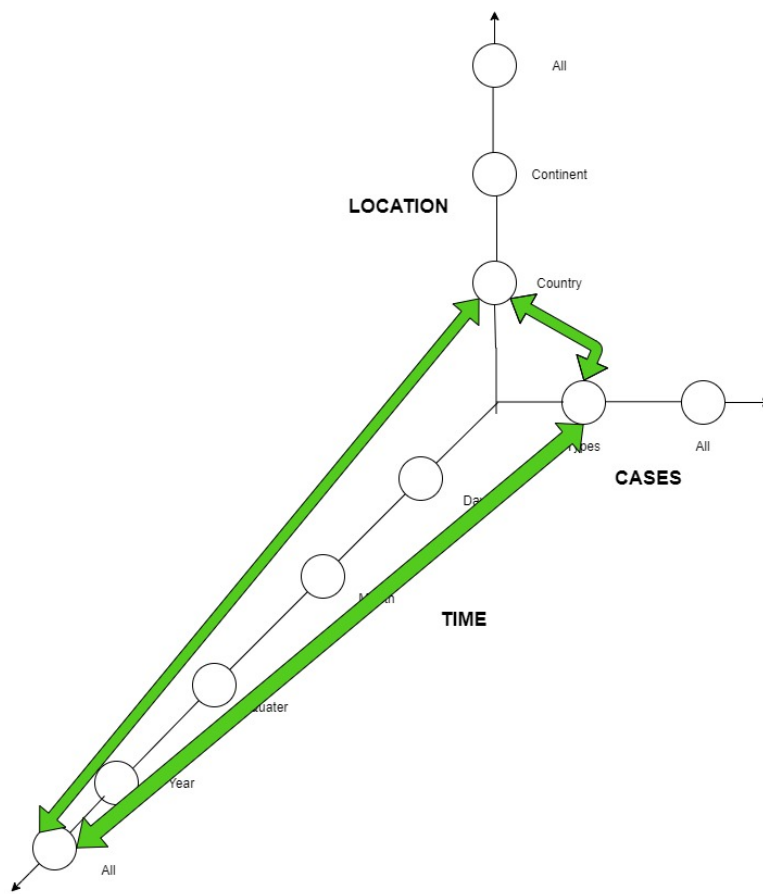


Figure 9

# 1.3 Database Schema

Our Database follows the structure of a Star Schema (Figure 10) containing Dim\_Case, Dim\_Location and Dim\_Time as dimension tables and Fact\_Covid\_effect as the fact table, containing the measure: 'people', which represents the no. of cases reported in accordance to the hierarchies of Case, Time and Location used.

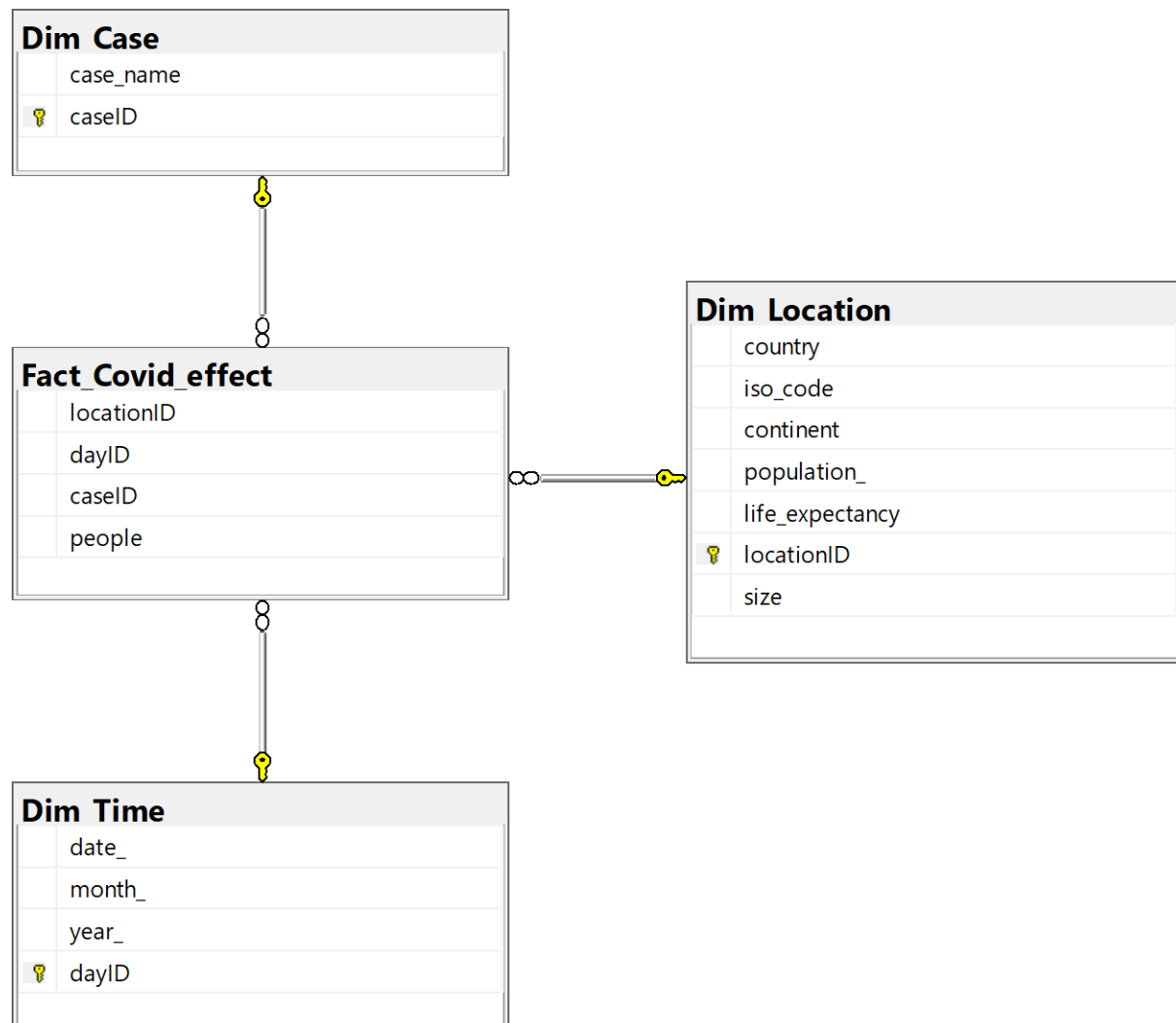


Figure 10- Star Schema

# 1.4 ETL Process

An extensive data cleaning process was used for clearing out irrelevant data and modifying the provided-useful data to a more usable format to be inserted into the database created via SQLMS. The ELT process for this project was conducted using Python (Jupyter Notebook) and Excel.

The ELT process is shown in detail below, in the order of creating the data to be inserted into the dimension tables and the data to be inserted into the fact tables.

## 1.4.1 Dimension tables:

### 1. Dim\_Location:

```
In [1]: import pandas as pd
import numpy as np

In [2]: data=pd.read_excel('owid-covid-data.xlsx')
data

In [3]: data.info()

In [4]: dimLocation = data[["location","iso_code","continent","population","life_expectancy"]]
dimLocation
dimLocation.info()

In [5]: #print(train[train.isnull().any(axis=1)][null_columns].head())
dimLocation[dimLocation["iso_code"].isnull()]

In [6]: dimLocation = dimLocation.dropna() #drops all null

In [7]: dimLocation= dimLocation.drop_duplicates(subset=['location']) #removes duplicate rows
dimLocation

In [8]: dimLocation["locationID"] = np.arange(1,len(dimLocation)+1)
print(dimLocation[0:1000])
```

```

In [9]: dimLocation=dimLocation.reset_index(drop=True) #assigns proper index
        dimLocation

...

In [13]: dimLocation['Size']=np.nan #Adds new empty column

In [14]: #small<=2000000
        #Large>=40000000

        for i in range( len(dimLocation)):
            if(int(dimLocation.population[i])<=2000000.0):
                dimLocation["Size"][i] = "Small"
            elif(int(dimLocation.population[i])>=40000000.0):
                dimLocation["Size"][i] = "Large"
            else:
                dimLocation["Size"][i] = "Medium"

        dimLocation

...

In [15]: dimLocation.to_csv(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Dimensions\dim_location2.csv', index = False, header=False)

```

Figure 11

The python script as shown in [Figure 11] was used in the data cleaning of the ‘owid-covid-data.xlsx’ data. Pandas was used to separate related columns such as: Location (country name), Continent, iso\_code, population and life expectancy.

The extracted columns were then added to a new table and 2 new columns called ‘locationID’ (surrogate key) and ‘Size’ were generated. ‘locationID’ was filled via a code to generate consecutive no.s and a for loop was run to populate the column of ‘Size’ with appropriate sizes according to the country’s population.

The table is then exported as a CSV file with no headers (&loaded into Dim\_Location in MSSMS) : ‘dim\_location’

## 2.Dim\_Cases:

```

In [2]: import pandas as pd
        import numpy as np

In [18]: dimCases = [{"recovered",1}, {"confirmed",2}, {"death",3}]
        dimCases=pd.DataFrame(dimCases,columns=["case_type","case_id"])
        dimCases

Out[18]:
   case_type  case_id
0  recovered         1
1  confirmed         2
2    death          3

In [19]: dimCases.to_csv(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Dimensions\dim_cases.csv', index = False, header=False) #

In [ ]:

```

Figure 12

The python script as shown in [Figure 12] was used to create a table to be loaded into ‘Dim\_Cases’ in MSSMS. The table contained the columns: ‘case\_type’ with fields of ‘recovered’, ‘confirmed’ and ‘death’; corresponding to which there’s a created: ‘case\_id’, containing consecutive no.s to be used as surrogate keys.

The table is then exported as a CSV file with no headers : ‘dim\_cases’.

### 3.Dim\_Time:

```
In [1]: import pandas as pd
import numpy as np

In [2]: data= pd.read_csv('time_series_covid19_confirmed_global.csv') #read the csv file
data
...
```

```
In [3]: data=data.drop(['Lat','Long','Province/State','Country/Region'],axis=1) #deletes unwanted columns

In [4]: #dim_time = pd.DataFrame(columns=["Date", "Month", "dayID"], dtype=[np.datetime64,np.object,np.int64])
dim_time = pd.DataFrame(columns=["Date", "Month", "Year", "dayID"]) #creates a new dataframe and adds columns

In [5]: data_T=data.T #Transpose data
data_T
...
```

```
In [6]: dim_time["Date"]=data_T.index #Adds value to column "date"
dim_time["Date"]=dim_time["Date"].astype('datetime64[ns]')

In [7]: #forloop to add values under 'months' and 'year'

months = {"1":"January","2":"February","3":"March","4":"April","5":"May","6":"June","7":"July","8":"August","9":"September","10":
for i in range(len(dim_time)):
    date_value = dim_time.Date[i]
    for key in months:
        if (key == str(date_value[date_value.find('/')])):
            dim_time.Month[i] = months[key]
            if(str(date_value[-2:])=="20"):
                dim_time.Year[i] = 2020
            else:
                dim_time.Year[i] = 2021
dim_time.tail()
< ... >
```

```
In [8]: dim_time["dayID"] = np.arange(1,len(dim_time)+1) #Assigns a dayID(for a surrogate key)

In [9]: dim_time
...
```

```
In [10]: dim_time["Date"]=dim_time["Date"].astype('datetime64[ns]') #Makes the date column to date-time format

In [11]: dim_time
...
```

```
In [14]: dim_time.to_csv (r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Dimensions\dim_time.csv', index = False, header=False) ###
< ... >
```

Figure 13

The python script as shown in [Figure 13] was used to create a table to be loaded into ‘Dim\_Time’ in MSSMS. Data is read from the file: ‘time\_series\_covid19\_confirmed\_global.csv’ into the data frame: ‘data’.

- Followed by which the columns of 'Country/Region', 'Lat', 'Long' and 'Province/State' are dropped.
- A new data frame called dim\_time is created with the columns of: 'Date', 'Month', 'Year' and 'dayID'.
- 'data' is transposed to allow for individual dates to form columns and forms data frame: 'date\_T'.
- A for loop is run:
  - to populate 'Date'(under 'dim\_time')with the individual dates from 'date\_T'.
  - to populate 'Month'(under 'dim\_time') with respective month titles of the dates according to the dictionary created above.
  - To populate the 'Year' (under 'dim\_time') with either 2020 or 2021 according to the last 2 characters of the dates in 'date\_T'
- 'Date' from 'dim\_time' is then converted to the date-time format.
- 'dim\_time' is exported as a CSV file with no headers: 'dim\_time'.

## 2.Fact Table data:

### 2.1 Fact\_confirmed:

```
import pandas as pd
import numpy as np

data=pd.read_csv('time_series_covid19_confirmed_global.csv')
|

data=data.groupby('Country/Region',as_index=False).sum()

data=data.drop(["Lat","Long"],axis=1)
data

d=data.T

cases_confirmed = pd.DataFrame(columns=["Country","Date","Count"])
cases_confirmed

#populates the new table - cases_confirmed with the country name corresponding to each date, corresponding to the no. of covid cases on the date
data_colNames = data.drop(["Country/Region"],axis=1).columns
for i in range(len(data)):
    x=0
    for j in range (len(data_colNames)):
        cases_confirmed=cases_confirmed.append({"Country":data["Country/Region"][i],"Date":data_colNames[j],"Count":(d.loc[data_colNames[j]][i])-x,ignore_index=True})
        x=d.loc[data_colNames[j]][i]

cases_confirmed["locID"] = np.nan #created new columns for foreign keys
cases_confirmed["dateID"] = np.nan
cases_confirmed["caseID"] = np.nan

cases_confirmed['Date']= cases_confirmed['Date'].astype('datetime64[ns]') #Makes the date column to date-time format

cases_confirmed=cases_confirmed[["Country","Date","locID","dateID","caseID","Count"]] #modify order of columns (for V-lookup in excel)
cases_confirmed

cases_confirmed.to_excel(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\confirmedX2.xlsx', index = False, header=False) #export as excel file for V-look up
cases_conf= pd.read_excel(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\confirmedX2.xlsx',header=None)
cases_conf[2] = cases_conf[2].values.astype(np.int64) #converts float value to int
cases_conf

cases_conf=cases_conf.drop([0,1],axis=1) #removes country name and date column

indexNames =cases_conf[ cases_conf[2] < 0 ].index #removes faulty values
cases_conf.drop(indexNames , inplace=True)

cases_conf.to_csv (r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\fact_confirmed2.csv', index = False, header=False) #### export as csv
```

Figure 14

The python script as shown in [Figure-14] was used to create the table containing data on confirmed cases:

- Data is read from the file: 'time\_series\_covid19\_confirmed\_global.csv' into the data frame: 'data'.
- The measures are summed up with grouping by the 'Country/Region' providing the data of each Country at each date.
- The columns of 'Lat' and 'Long' are dropped and the data is transposed to form a new data frame: 'd'.
- A new data frame called 'cases\_confirmed' is created with columns of: 'Country', 'Date' and 'Count'.
- A for loop is then run populating 'cases\_confirmed' with the data of each country, corresponding to every date, corresponding to the measure of the case at each date.

- New columns of 'locID', 'dateID' (later converted to date-time format) and 'caseID' are created and all the columns of cases\_confirmed are re-ordered to be used for Vlookup in excel.
- 'cases\_confirmed' is exported as an excel file.
- Vlook up is performed for each of the columns of 'locID', 'dateID' and 'caseID' using respective dimension table CSV files (such as that of 'dim\_location', 'dim\_time' and 'dim\_cases'). An example of this process is shown below in [Figure-14] (extracting of 'locationID' using 'dim\_time' csv file).
- The modified excel file of 'cases\_confirmed' (containing data of 'locID', 'dateID' and 'caseID') is read using pandas to 'cases\_conf'
- 'cases\_conf' drops the columns 0 & 1 which contain data of each country name and the individual dates and corrects the data type of column 2 to int.
- A data anomaly which contained negative numbers was noted and these data were removed.
- 'cases\_rec' is then finally exported as a CSV file: 'fact\_confirmed2.csv'.

The image displays two Excel spreadsheets side-by-side, illustrating the VLOOKUP process for extracting location IDs.

**Left Spreadsheet (dim\_location):**

	A	B	C	D	E	F	G	H	I	J	K
1	Afghanistan	AFG	Asia	38928341	64.83	1	Medium				
2	Albania	ALB	Europe	2877800	78.57	2	Medium				
3	Algeria	DZA	Africa	43851043	76.88	3	Large				
4	Andorra	AND	Europe	77265	83.73	4	Small				
5	Angola	AGO	Africa	32866268	61.15	5	Medium				
6	Antigua and Barbuda	ATG	North America	97928	77.02	6	Small				
7	Argentina	ARG	South America	45195777	76.67	7	Large				
8	Armenia	ARM	Asia	2963234	75.09	8	Medium				
9	Australia	AUS	Oceania	25499881	83.44	9	Medium				
10	Austria	AUT	Europe	9006400	81.54	10	Medium				
11	Azerbaijan	AZE	Asia	10139175	73	11	Medium				
12	Bahamas	BHS	North America	393248	73.92	12	Small				
13	Bahrain	BHR	Asia	1701583	77.29	13	Small				
14	Bangladesh	BGD	Asia	1.65E+08	72.59	14	Large				
15	Barbados	BRB	North America	287371	79.19	15	Small				
16	Belarus	BLR	Europe	9449321	74.79	16	Medium				
17	Belgium	BEL	Europe	11589616	81.63	17	Medium				
18	Belize	BLZ	North America	397621	74.62	18	Small				
19	Benin	BEN	Africa	12123198	61.77	19	Medium				
20	Bhutan	BTN	Asia	771612	71.78	20	Small				
21	Bolivia	BOL	South America	11673029	71.51	21	Medium				
22	Bosnia and Herzegovina	BIH	Europe	3280815	77.4	22	Medium				
23	Botswana	BWA	Africa	2351625	69.59	23	Medium				
24	Brazil	BRA	South America	2.13E+08	75.88	24	Large				
25	Brunei	BRN	Asia	437483	75.86	25	Small				
26	Bulgaria	BGR	Europe	6948445	75.05	26	Medium				
27	Burkina Faso	BFA	Africa	20903278	61.58	27	Medium				
28	Burundi	BDI	Africa	11890781	61.58	28	Medium				
29	Cambodia	KHM	Asia	16718971	69.82	29	Medium				

**Right Spreadsheet (confir...):**

	A	B	C	D	E	F	G	H	I	J
1	Afghanistan	2020-01-22 00:00:00	=VLOOKUP(A1;dim_location.csv!\$A\$1:\$G\$189;6;0)							
2	Afghanistan	2020-01-23 00:00:00								
3	Afghanistan	2020-01-24 00:00:00								
4	Afghanistan	2020-01-25 00:00:00								
5	Afghanistan	2020-01-26 00:00:00								
6	Afghanistan	2020-01-27 00:00:00								
7	Afghanistan	2020-01-28 00:00:00								
8	Afghanistan	2020-01-29 00:00:00								
9	Afghanistan	2020-01-30 00:00:00								
10	Afghanistan	2020-01-31 00:00:00								
11	Afghanistan	2020-02-01 00:00:00								
12	Afghanistan	2020-02-02 00:00:00								
13	Afghanistan	2020-02-03 00:00:00								
14	Afghanistan	2020-02-04 00:00:00								
15	Afghanistan	2020-02-05 00:00:00								
16	Afghanistan	2020-02-06 00:00:00								
17	Afghanistan	2020-02-07 00:00:00								
18	Afghanistan	2020-02-08 00:00:00								
19	Afghanistan	2020-02-09 00:00:00								
20	Afghanistan	2020-02-10 00:00:00								
21	Afghanistan	2020-02-11 00:00:00								
22	Afghanistan	2020-02-12 00:00:00								
23	Afghanistan	2020-02-13 00:00:00								
24	Afghanistan	2020-02-14 00:00:00								
25	Afghanistan	2020-02-15 00:00:00								
26	Afghanistan	2020-02-16 00:00:00								
27	Afghanistan	2020-02-17 00:00:00								
28	Afghanistan	2020-02-18 00:00:00								
29	Afghanistan	2020-02-19 00:00:00								

Figure 15-V-look



## 2.2 Fact\_recovered:

```
import pandas as pd
import numpy as np

data=pd.read_csv('time_series_covid19_recovered_global.csv')

data=data.groupby('Country/Region',as_index=False).sum()

data=data.drop(["Lat","Long"],axis=1)

cases_recovered = pd.DataFrame(columns=["Country","Date","Count"]) #removes the columns Lat and Long
cases_recovered

d=data.T #transposes data

#populates the new table - cases_recovered with the country name corresponding to each date, corresponding to the data reported on the date
data_colNames = data.drop(["Country/Region"],axis=1).columns
for i in range(len(data)):
    x=d
    for j in range (len(data_colNames)):
        cases_recovered=cases_recovered.append({"Country":data["Country/Region"][i],"Date":data_colNames[j],"Count":(d.loc[data_colNames[j]][i]-x),ignore_index=True})
    x=d.loc[data_colNames[j]][i]

cases_recovered

cases_recovered["locID"] = np.nan
cases_recovered["dateID"] = np.nan
cases_recovered["caseID"] = np.nan #created new columns for foreign keys

cases_recovered['Date']= cases_recovered['Date'].astype('datetime64[ns]') #Makes the date column to date-time format

cases_recovered=cases_recovered[['Country','Date','locID','dateID','caseID','Count']] #modify order of columns (for V-lookup in excel)

cases_recovered.to_excel(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\recoveredX2.xlsx', index = False, header=False) #export as excel file for V-look up

cases_rec= pd.read_excel(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\recoveredX2.xlsx',header=None)

cases_rec[2] = cases_rec[2].values.astype(np.int64) #converts float value to int

cases_rec=cases_rec.drop([0,1],axis=1) #removes country name and date column and leaves only the foreign keys

indexNames=cases_rec[ cases_rec[2] < 0 ].index #removes faulty values

cases_rec.drop(indexNames , inplace=True)

cases_rec.to_csv (r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\fact_recovered2.csv', index = False, header=False) #### export as csv
```

Figure 16

The python script as shown above in [Figure-16] was used to create the table containing data on recovered cases:

- Data is read from the file: 'time\_series\_covid19\_recovered\_global.csv' into the data frame: 'data'.
- The measures are summed up with grouping by the 'Country/Region' providing the data of each Country at each date.
- The columns of 'Lat' and 'Long' are dropped and the data is transposed to form a new data frame: 'd'.
- A new data frame called 'cases\_recovered' is created with columns of: 'Country', 'Date' and 'Count'.
- A for loop is then run populating 'cases\_recovered' with the data of each country, corresponding to every date, corresponding to the measure of the case at each date.
- New columns of 'locID', 'dateID' (later converted to date-time format) and 'caseID' are created and all the columns of 'cases\_recovered' are re-ordered to be used for V-lookup in excel.
- 'cases\_recovered' is exported as an excel file.

- V-look up is performed for each of the columns of ‘locID’, ‘dateID’ and ‘caseID’ using respective dimension table CSV files (such as that of ‘dim\_location’, ‘dim\_time’ and ‘dim\_cases’). An example of this process is shown below in [Figure-17] (extracting of timeID using ‘dim\_time’ csv file).
- The modified excel file of ‘cases\_recovered’ (containing data of ‘locID’, ‘dateID’ and ‘caseID’) is read using pandas to ‘cases\_rec’
- ‘cases\_rec’ drops the columns 0 & 1 which contain data of each country name and the individual dates.
- A data anomaly which contained negative numbers was noted and these data were removed.
- ‘cases\_rec’ is then finally exported as a CSV file: ‘fact\_recovered2.csv’.

The figure consists of two side-by-side screenshots of the Microsoft Excel interface. The left screenshot shows a spreadsheet with columns A through K. Column A contains dates from 22/01/2020 to 19/02/2020. Column B contains the year 2020. Column C contains a sequence of numbers from 1 to 29. The formula bar shows the formula '=VLOOKUP(B1;dim\_time.csv!\$A\$1:\$D\$405;4;0)'. The right screenshot shows a similar spreadsheet, but with the data from the CSV file loaded. Column A now contains the country name 'Afghanistan' and the date '2020-01-22 00:00:00'. Column B contains the year '2020'. Column C contains the sequence of numbers from 1 to 29. The formula bar shows the same formula. The spreadsheet is titled 'Sheet1'.

	A	B	C	D	E	F	G	H	I	J	K
1	22/01/2020	January	2020	1							
2	23/01/2020	January	2020	2							
3	24/01/2020	January	2020	3							
4	25/01/2020	January	2020	4							
5	26/01/2020	January	2020	5							
6	27/01/2020	January	2020	6							
7	28/01/2020	January	2020	7							
8	29/01/2020	January	2020	8							
9	30/01/2020	January	2020	9							
10	31/01/2020	January	2020	10							
11	01/02/2020	February	2020	11							
12	02/02/2020	February	2020	12							
13	03/02/2020	February	2020	13							
14	04/02/2020	February	2020	14							
15	05/02/2020	February	2020	15							
16	06/02/2020	February	2020	16							
17	07/02/2020	February	2020	17							
18	08/02/2020	February	2020	18							
19	09/02/2020	February	2020	19							
20	10/02/2020	February	2020	20							
21	11/02/2020	February	2020	21							
22	12/02/2020	February	2020	22							
23	13/02/2020	February	2020	23							
24	14/02/2020	February	2020	24							
25	15/02/2020	February	2020	25							
26	16/02/2020	February	2020	26							
27	17/02/2020	February	2020	27							
28	18/02/2020	February	2020	28							
29	19/02/2020	February	2020	29							

	A	B	C	D	E	F	G	H	I	J
1	Afghanistan	2020-01-22 00:00:00	1	1	1	0				
2	Afghanistan	2020-01-23 00:00:00	1	2	1	0				
3	Afghanistan	2020-01-24 00:00:00	1	3	1	0				
4	Afghanistan	2020-01-25 00:00:00	1	4	1	0				
5	Afghanistan	2020-01-26 00:00:00	1	5	1	0				
6	Afghanistan	2020-01-27 00:00:00	1	6	1	0				
7	Afghanistan	2020-01-28 00:00:00	1	7	1	0				
8	Afghanistan	2020-01-29 00:00:00	1	8	1	0				
9	Afghanistan	2020-01-30 00:00:00	1	9	1	0				
10	Afghanistan	2020-01-31 00:00:00	1	10	1	0				
11	Afghanistan	2020-02-01 00:00:00	1	11	1	0				
12	Afghanistan	2020-02-02 00:00:00	1	12	1	0				
13	Afghanistan	2020-02-03 00:00:00	1	13	1	0				
14	Afghanistan	2020-02-04 00:00:00	1	14	1	0				
15	Afghanistan	2020-02-05 00:00:00	1	15	1	0				
16	Afghanistan	2020-02-06 00:00:00	1	16	1	0				
17	Afghanistan	2020-02-07 00:00:00	1	17	1	0				
18	Afghanistan	2020-02-08 00:00:00	1	18	1	0				
19	Afghanistan	2020-02-09 00:00:00	1	19	1	0				
20	Afghanistan	2020-02-10 00:00:00	1	20	1	0				
21	Afghanistan	2020-02-11 00:00:00	1	21	1	0				
22	Afghanistan	2020-02-12 00:00:00	1	22	1	0				
23	Afghanistan	2020-02-13 00:00:00	1	23	1	0				
24	Afghanistan	2020-02-14 00:00:00	1	24	1	0				
25	Afghanistan	2020-02-15 00:00:00	1	25	1	0				
26	Afghanistan	2020-02-16 00:00:00	1	26	1	0				
27	Afghanistan	2020-02-17 00:00:00	1	27	1	0				
28	Afghanistan	2020-02-18 00:00:00	1	28	1	0				
29	Afghanistan	2020-02-19 00:00:00	1	29	1	0				

Figure 17-V-lookup

## 2.3 Fact\_deaths:

```
import pandas as pd
import numpy as np

data=pd.read_csv('time_series_covid19_deaths_global.csv')

data=data.groupby('Country/Region',as_index=False).sum()

data=data.drop(["Lat","Long"],axis=1)

cases_deaths = pd.DataFrame(columns=["Country","Date","Count"]) #creates new dataframe
cases_deaths

d=data.T #transposes the data

#populates the new table - cases_recovered with the country name corresponding to each date, corresponding to the data reported on the date
data_colNames = data.drop(["Country/Region"],axis=1).columns
for i in range(len(data)):
    x=0
    for j in range (len(data_colNames)):
        cases_deaths =cases_deaths.append({"Country":data["Country/Region"][i],"Date":data_colNames[j],"Count":(d.loc[data_colNames[j]][i])-x,ignore_index=True)}
        x=d.loc[data_colNames[j]][i]

cases_deaths["locID"] = np.nan #created new columns for foreign keys
cases_deaths["dateID"] = np.nan
cases_deaths["caseID"] = np.nan

cases_deaths['Date']= cases_deaths['Date'].astype('datetime64[ns]') #Makes the date column to date-time format

cases_deaths= cases_deaths[['Country','Date','locID','dateID','caseID','Count']] #modify order of columns (for V-lookup in excel)

cases_deaths.to_excel(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\deathsX2.xlsx', index = False, header=False) #export as excel file for V-look up
cases_deaths

cases_d= pd.read_excel(r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\deathsX2.xlsx',header=None)

cases_d[2] = cases_d[2].values.astype(np.int64) #converts float value to int

cases_d=cases_d.drop([0,1],axis=1) #Removes country name and date column

indexNames =cases_d[ cases_d[2] < 0 ].index

cases_d.drop(indexNames , inplace=True)

cases_d.to_csv (r'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\fact_deaths2.csv', index = False, header=False) #### export as csv
```

Figure 18

The python script as shown above in [Figure 18] was used to create the table containing data on recovered cases:

- Data is read from the file: 'time\_series\_covid19\_deaths\_global.csv' into the data frame: 'data'.
- The measures are summed up with grouping by the 'Country/Region' providing the data of each Country at each date.
- The columns of 'Lat' and 'Long' are dropped and the data is transposed to form a new data frame: 'd'.
- A new data frame called 'cases\_deaths' is created with columns of: 'Country', 'Date' and 'Count'.
- A for loop is then run populating 'cases\_deaths' with the data of each country, corresponding to every date, corresponding to the measure of the case at each date.
- New columns of 'locID', 'dateID' (later converted to date-time format) and 'caseID' are created and all the columns of 'cases\_deaths' are re-ordered to be used for V-lookup in excel.

- ‘cases\_deaths’ is exported as an excel file.
- V-look up is performed for each of the columns of ‘locID’, ‘dateID’ and ‘caseID’ using respective dimension table CSV files (such as that of ‘dim\_location’, ‘dim\_time’ and ‘dim\_cases’). An example of this process is shown below in *[Figure 19]* (extracting of timeID using ‘dim\_time’ csv file).
- The modified excel file of ‘cases\_recovered’ (containing data of ‘locID’, ‘dateID’ and ‘caseID’) is read using pandas to ‘cases\_d’
- ‘cases\_d’ drops the columns 0 & 1 which contain data of each country name and the individual dates.
- Anomalous negative values are noted in ‘cases\_d’, therefore these values are deleted
- ‘cases\_d’ is then finally exported as a CSV file: ‘fact\_deaths2.csv’.

The figure consists of two side-by-side screenshots of the Microsoft Excel interface. Both windows are titled 'Han Vignesh Amirthalingam (22874425)'. The left window has a file named 'dim\_time...' and shows a table with columns A through K. The data in columns A and B consists of dates from 22/01/2020 to 19/02/2020, with column C containing the month and column D containing the year. The formula bar shows '=VLOOKUP(B1;dim\_time.csv!\$A\$1:\$D\$405;4;0)'. The right window has a file named 'deathX2...' and shows a table with columns A through H. The data in column A consists of country names (Afghanistan) and in column B consists of dates from 2020-01-22 00:00:00 to 2020-02-19 00:00:00. The formula bar shows '=VLOOKUP(B1;dim\_time.csv!\$A\$1:\$D\$405;4;0)'. Both tables have data in columns C, D, E, and F, which appear to be numerical values.

	A	B	C	D	E	F	G	H
1	22/01/2020	January	2020	1				
2	23/01/2020	January	2020	2				
3	24/01/2020	January	2020	3				
4	25/01/2020	January	2020	4				
5	26/01/2020	January	2020	5				
6	27/01/2020	January	2020	6				
7	28/01/2020	January	2020	7				
8	29/01/2020	January	2020	8				
9	30/01/2020	January	2020	9				
10	31/01/2020	January	2020	10				
11	01/02/2020	February	2020	11				
12	02/02/2020	February	2020	12				
13	03/02/2020	February	2020	13				
14	04/02/2020	February	2020	14				
15	05/02/2020	February	2020	15				
16	06/02/2020	February	2020	16				
17	07/02/2020	February	2020	17				
18	08/02/2020	February	2020	18				
19	09/02/2020	February	2020	19				
20	10/02/2020	February	2020	20				
21	11/02/2020	February	2020	21				
22	12/02/2020	February	2020	22				
23	13/02/2020	February	2020	23				
24	14/02/2020	February	2020	24				
25	15/02/2020	February	2020	25				
26	16/02/2020	February	2020	26				
27	17/02/2020	February	2020	27				
28	18/02/2020	February	2020	28				
29	19/02/2020	February	2020	29				

	A	B	C	D	E	F	G	H
1	Afghanistan	2020-01-22 00:00:00	1	1	3	0		
2	Afghanistan	2020-01-23 00:00:00	1	2	3	0		
3	Afghanistan	2020-01-24 00:00:00	1	3	3	0		
4	Afghanistan	2020-01-25 00:00:00	1	4	3	0		
5	Afghanistan	2020-01-26 00:00:00	1	5	3	0		
6	Afghanistan	2020-01-27 00:00:00	1	6	3	0		
7	Afghanistan	2020-01-28 00:00:00	1	7	3	0		
8	Afghanistan	2020-01-29 00:00:00	1	8	3	0		
9	Afghanistan	2020-01-30 00:00:00	1	9	3	0		
10	Afghanistan	2020-01-31 00:00:00	1	10	3	0		
11	Afghanistan	2020-02-01 00:00:00	1	11	3	0		
12	Afghanistan	2020-02-02 00:00:00	1	12	3	0		
13	Afghanistan	2020-02-03 00:00:00	1	13	3	0		
14	Afghanistan	2020-02-04 00:00:00	1	14	3	0		
15	Afghanistan	2020-02-05 00:00:00	1	15	3	0		
16	Afghanistan	2020-02-06 00:00:00	1	16	3	0		
17	Afghanistan	2020-02-07 00:00:00	1	17	3	0		
18	Afghanistan	2020-02-08 00:00:00	1	18	3	0		
19	Afghanistan	2020-02-09 00:00:00	1	19	3	0		
20	Afghanistan	2020-02-10 00:00:00	1	20	3	0		
21	Afghanistan	2020-02-11 00:00:00	1	21	3	0		
22	Afghanistan	2020-02-12 00:00:00	1	22	3	0		
23	Afghanistan	2020-02-13 00:00:00	1	23	3	0		
24	Afghanistan	2020-02-14 00:00:00	1	24	3	0		
25	Afghanistan	2020-02-15 00:00:00	1	25	3	0		
26	Afghanistan	2020-02-16 00:00:00	1	26	3	0		
27	Afghanistan	2020-02-17 00:00:00	1	27	3	0		
28	Afghanistan	2020-02-18 00:00:00	1	28	3	0		
29	Afghanistan	2020-02-19 00:00:00	1	29	3	0		

Figure 19-V-look-up

# 1.5 Implementation

## 1.5.1 Building & Populating the Database

The database was created using the *sql\_script.sql* script, which is provided. This script creates and populates the required Dimension and Fact tables, as well as adds constraints for referencing primary keys and foreign keys.

### 1.5.1.1 Dimension tables

```
DROP TABLE IF EXISTS Dim_Time
GO

Create table Dim_Time
(
    date_ date,
    month_ varchar(50),
    year_ int,
    dayID int PRIMARY KEY IDENTITY(1,1)
)
Go

BULK INSERT Dim_Time FROM 'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Dimensions\dim_time.csv' ---bulk insert CSV data
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
```

Figure 20

```
DROP TABLE IF EXISTS Dim_Location
GO

Create table Dim_Location
(
    country varchar(50),
    iso_code varchar(50),
    continent varchar(50),
    population float,
    life_expectancy float,
    locationID int PRIMARY KEY IDENTITY(1,1),
    size varchar(50)
)
Go

BULK INSERT Dim_Location FROM 'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Dimensions\dim_location.csv' ---bulk insert CSV data
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
```

Figure 21

```

--
DROP TABLE IF EXISTS Dim_Case
GO

Create table Dim_Case
(
case_name varchar(50),
caseID int PRIMARY KEY IDENTITY(1,1)
)
Go

BULK INSERT Dim_Case FROM 'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Dimensions\dim_cases.csv' ---bulk insert CSV data
WITH (
CHECK_CONSTRAINTS,
--CODEPAGE='ACP',
DATAFILETYPE='char',
FIELDTERMINATOR=',',
ROWTERMINATOR='\n',
KEEPIDENTITY,
TABLOCK
);

```

Figure 22

[Figures 20,21,22] are the scripts used in the creation of the dimension tables: Dim\_Time, Dim\_Location and Dim\_Case. Each of these tables have a surrogate key column (eg : dayID, locationID and caseID), that would be used as a reference in the fact table. 'Create Table' is also followed by the 'BULK INSERT' command which inserts all the data present in the respective CSV files, into the dimension tables.

### 15.1.2 Fact tables

```

DROP TABLE IF EXISTS Fact_Covid_effect
GO

Create table Fact_Covid_effect
(
locationID int,
dayID int,
caseID int,
people int
)
Go

```

Figure 23

[Figure 23] is the script run to create the fact table : Fact\_Covid\_Effect. Where, attributes such as: 'locationID', 'dayID' and 'caseID' are forging keys that reference to the dimension tables; the attribute 'people' is the fact table measure.

This is followed by the script in [Figure 24] which sets the Foreign Key Constraints; and the script in [Figure 25] which 'BULK INSERTS' data from 3

csv files that contain data of confirmed (fact\_confirmed.csv), deaths (fact\_deaths.csv) and recovered (fact\_recovered.csv) cases.

```
---ADD CONSTRAINTS
ALTER TABLE Fact_Covid_effect ADD CONSTRAINT
FK_locationID FOREIGN KEY (locationID) REFERENCES Dim_Location(locationID);
ALTER TABLE Fact_Covid_effect ADD CONSTRAINT
FK_timeID FOREIGN KEY (dayID) REFERENCES Dim_Time(dayID);
ALTER TABLE Fact_Covid_effect ADD CONSTRAINT
FK_caseID FOREIGN KEY (caseID) REFERENCES Dim_Case(caseID);
Go
```

Figure 24

```
BULK INSERT Fact_Covid_effect FROM 'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\fact_confirmed2.csv' ---bulk insert CSV data
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY
    TABLOCK
);

BULK INSERT Fact_Covid_effect FROM 'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\fact_recovered2.csv' ---bulk insert CSV data
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY
    TABLOCK
);

BULK INSERT Fact_Covid_effect FROM 'C:\UNIVERSITY\Year 2\Sem 1\CITS3401\project1.0\Data\Fact tables\fact_deaths2.csv' ---bulk insert CSV data
WITH (
    CHECK_CONSTRAINTS,
    --CODEPAGE='ACP',
    DATAFILETYPE='char',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    --KEEPIDENTITY
    TABLOCK
);
```

Figure 25

## 2 Usage & Visualisation

The database created could be used to get results for the listed business queries via the use of SSMS, Visual Studio and PowerBI, by using SSAS tools. The business queries are listed below along with their answer visualisations and OLAP operations.

### 1. Queries:

- a. What is the total number of confirmed cases in Australia in 2020?

OLAP:

- The data cube is drilled down in the time dimension to year.
- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for year=2020, country=Australia and case=confirmed

POWER BI:

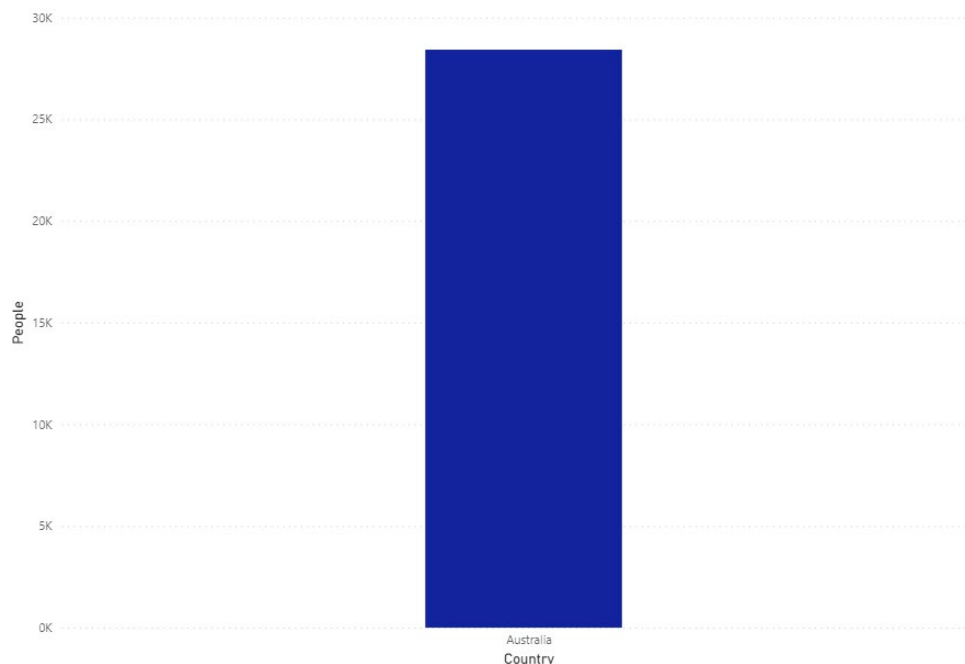


Figure 26

Answer: 28425

- b. What is the number of confirmed cases in each quarter of 2020 in Australia?



## OLAP:

- The data cube is drilled down in the time dimension to Month.
- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for case=confirmed ; country=Australia and time = (Jan, Feb, Mar), (Apr, May, Jun), (Jul, Aug, Sept) and (Oct, Nov, Dec) , each set of three months, representing a quarter.

## POWER BI:

### Quarter1: Jan-March

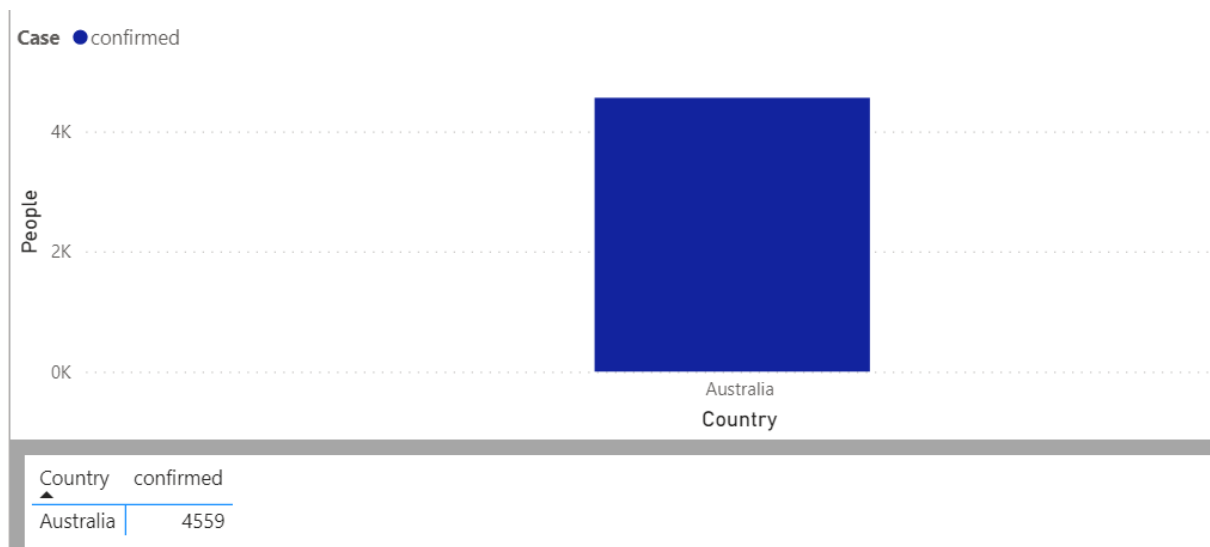


Figure 27

### Quarter2:Apr-Jun

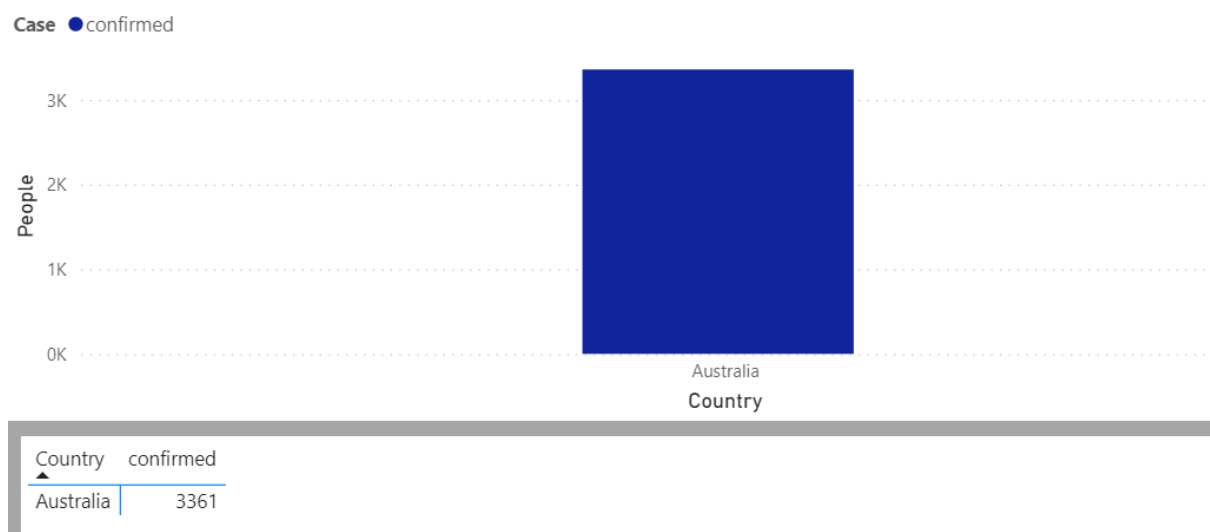


Figure 28

### Quarter3: Jul-Sept

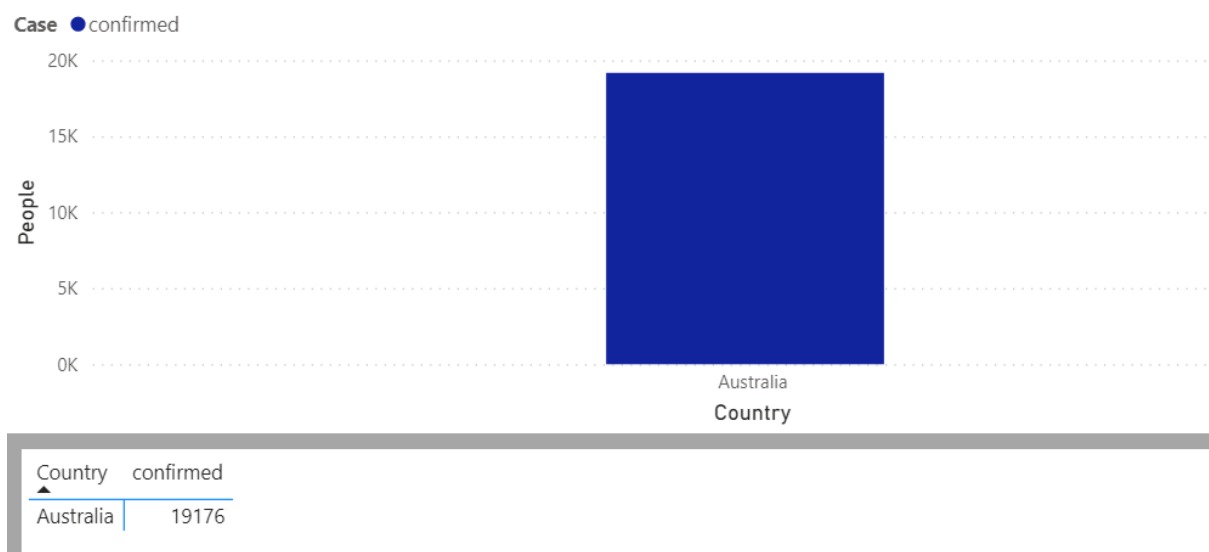


Figure 29

### Quarter4: Oct-Dec

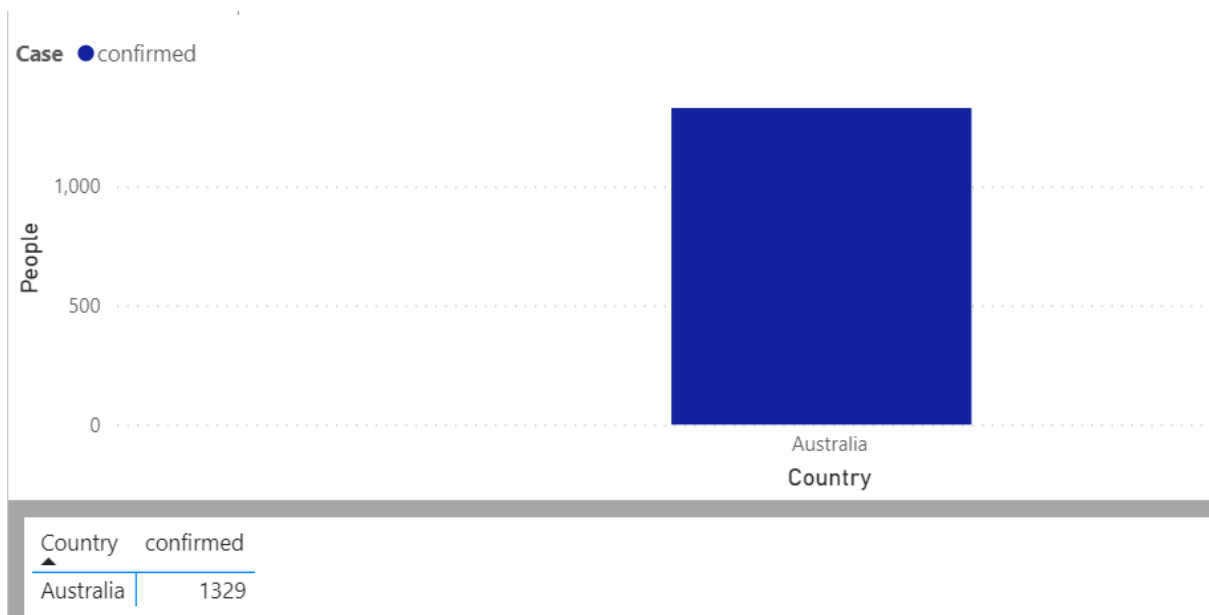


Figure 30

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

OLAP:

- The data cube is drilled down in the time dimension to Month.
- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for case=confirmed; country=Australia and time = all the months.

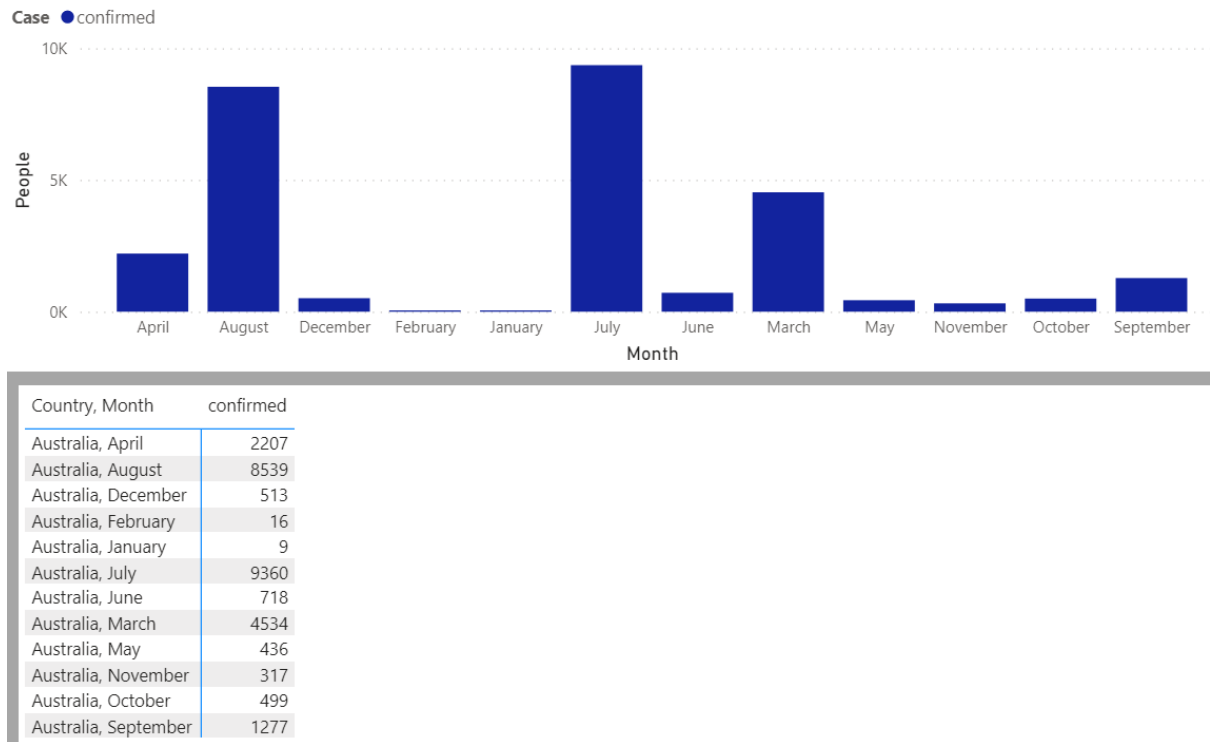


Figure 31

## 2. Queries:

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

## OLAP:

- The data cube is drilled down in the time dimension to Year.
- The data cube is drilled down in the location dimension to Continent.
- The data cube is then diced for case=recovered ; Continent= North America & South America
- Drill up the values to aggregate and find the recovered cases for 2020 as a combination of both Americas

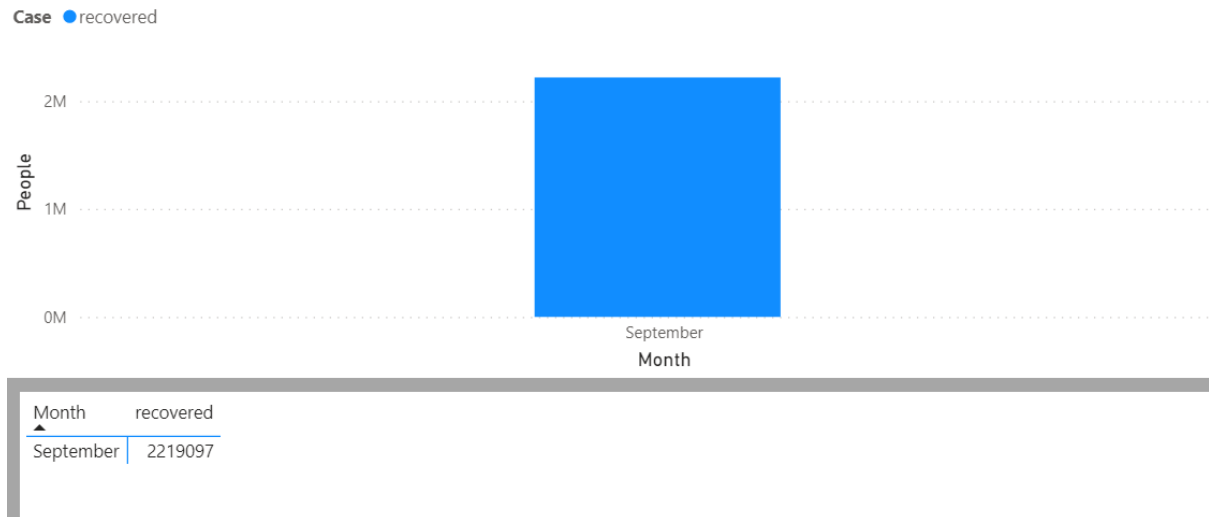


Figure 32

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

OLAP:

- The data cube is drilled down in the time dimension to Year= 2020
- Drilled down to Month.
- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for case=recovered ; country= United States, Canada and Mexico; Month = Sept

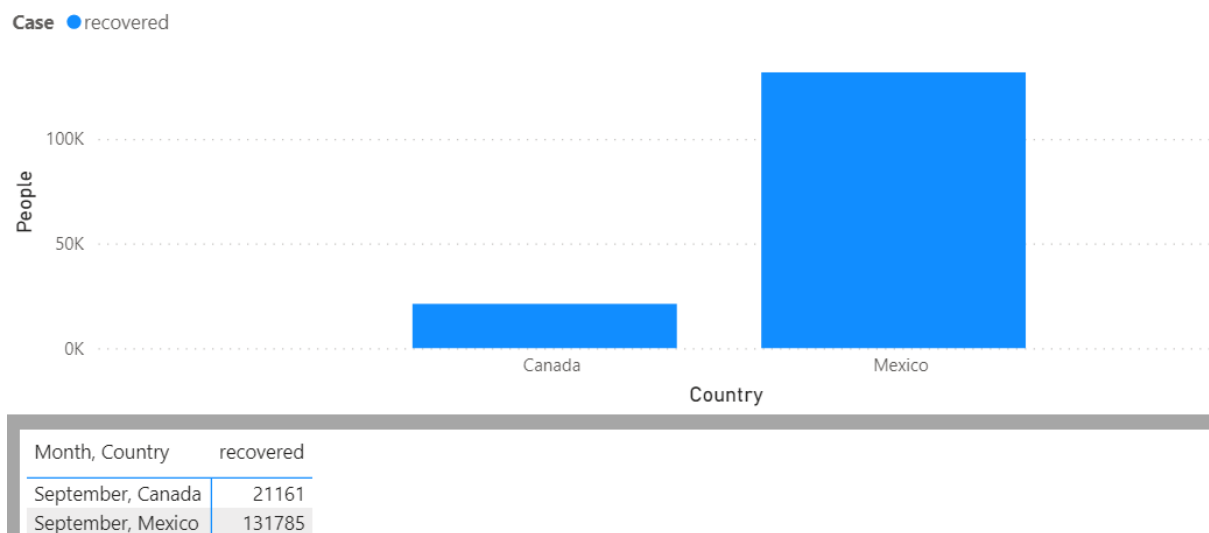


Figure 33

### 3. Queries:

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

OLAP:

- The data cube is drilled up to ALL in Time dimension
- The data cube is drilled down in the location dimension to ALL
- The data cube is then sliced for Case= death

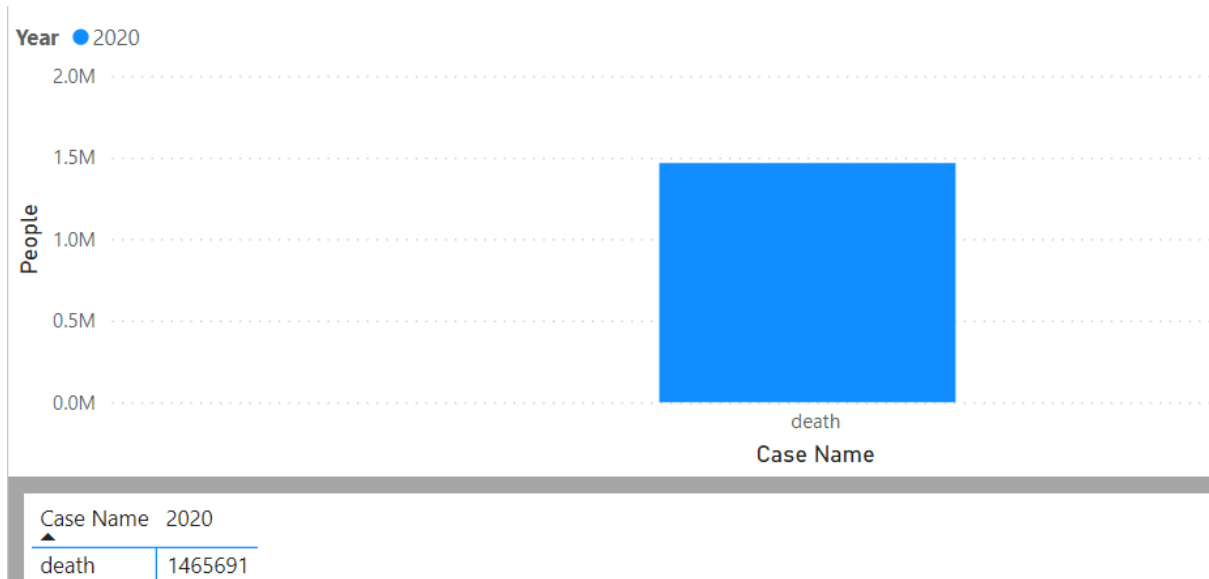


Figure 34

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

OLAP:

- The data cube is drilled down in the time dimension to Year= 2020
- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for Case=death ; country= 'Large', 'Small' and 'Medium'; Year = 2020

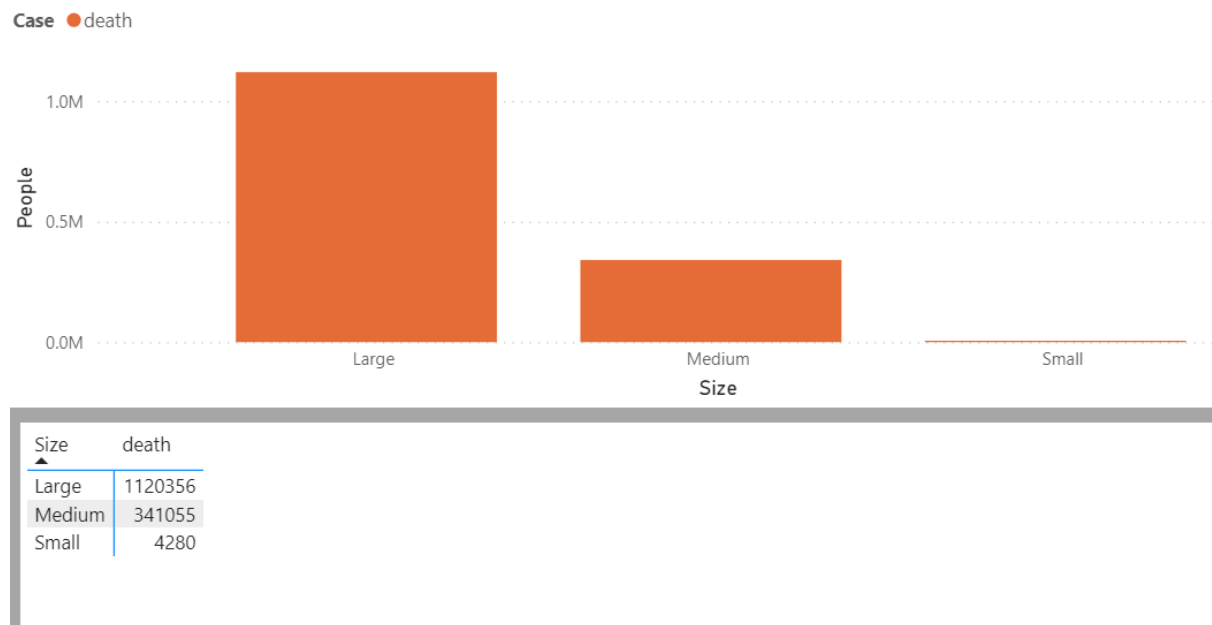


Figure 35

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

OLAP:

- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for case=recovered; country is grouped by life expectancy > 75; Year = 2020 and 2021
- The data cube is then diced for case=confirmed; country is grouped by life expectancy > 75; Year = 2020 and 2021

## Recovered and confirmed cases of countries with life expectancy > 75:

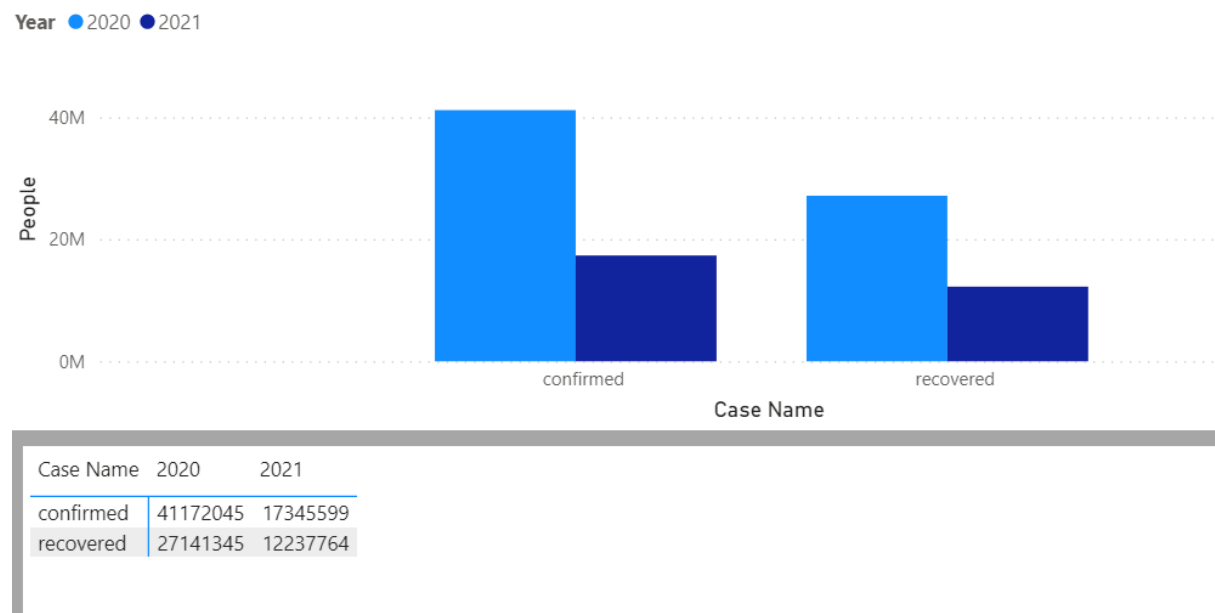


Figure 36

## OLAP:

- The data cube is drilled up in the location dimension to ALL
- The data cube is drilled down in the location dimension to country.
- The data cube is then diced for case=recovered ; Year = 2020 and 2021

## Recovered and confirmed cases of all countries:

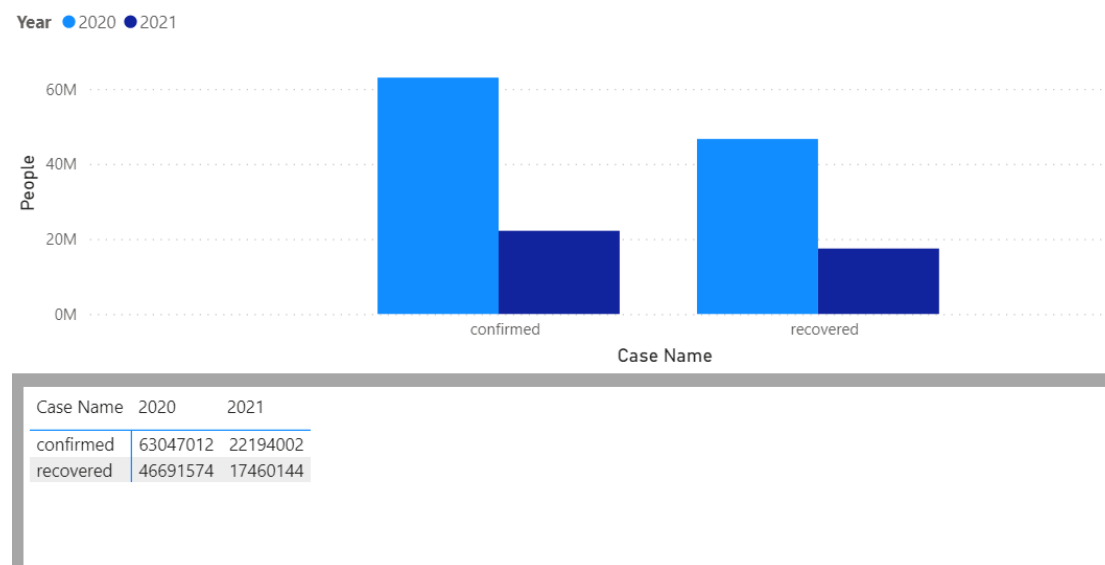


Figure 37

For 2020 & 2021:

	Life expectancy > 75	Life expectancy <=75
Recovered Cases	39379109	24772609
Confirmed Cases	58517644	26723370
Recovery rate = (Recovered cases/ Confirmed cases)	0.67	0.92

Conclusion: The recovery rate shows that countries with life expectancy > 75 does not have a higher recovery rate.