CITS3401 Data Warehousing
Project 1 Report

Hari Vignesh Amirthalingam (22874425)

Semester 1 2021

1. Design and Implementation

1.1 Concept Hierarchies & StarNet Model

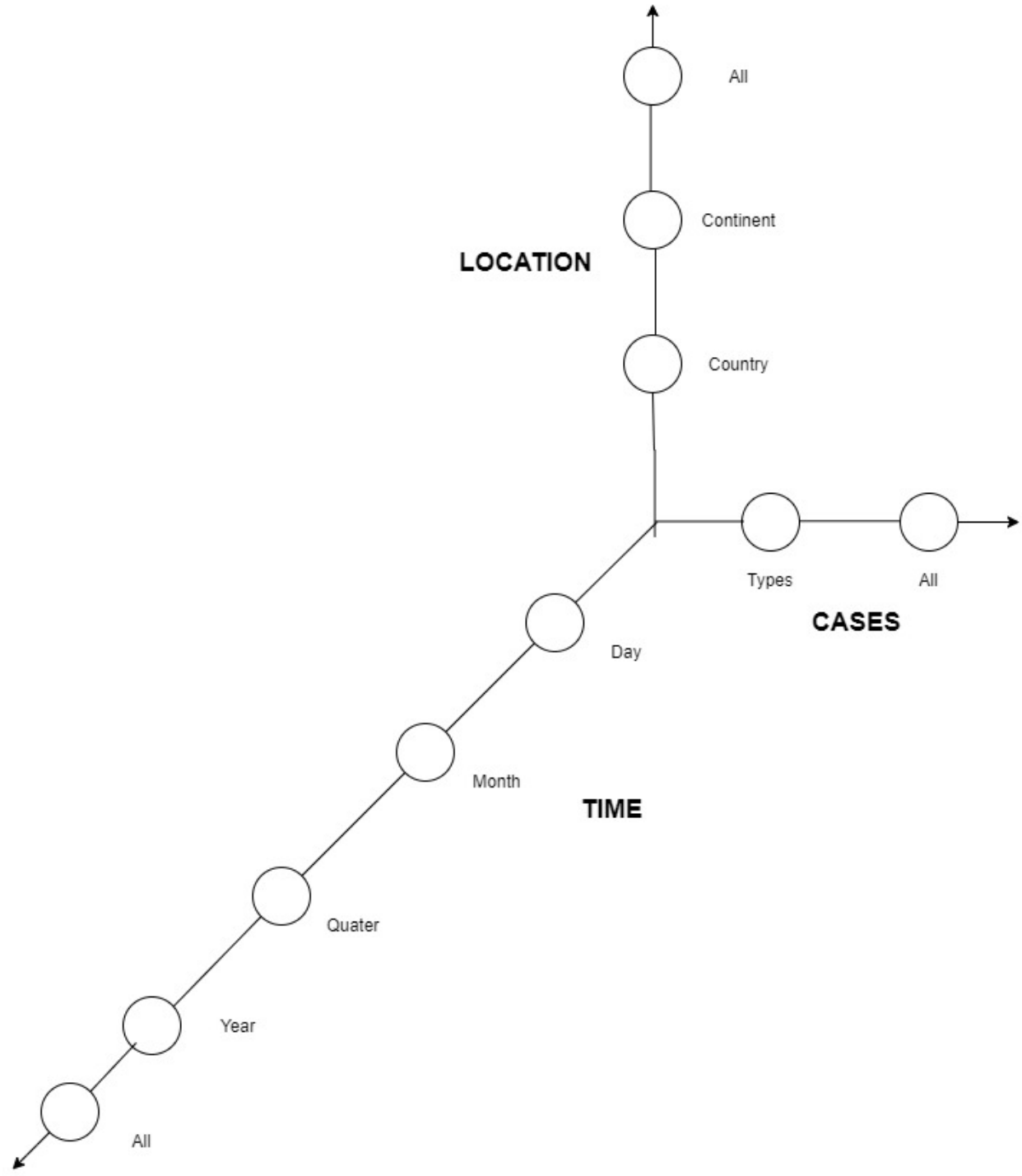


Figure 1

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

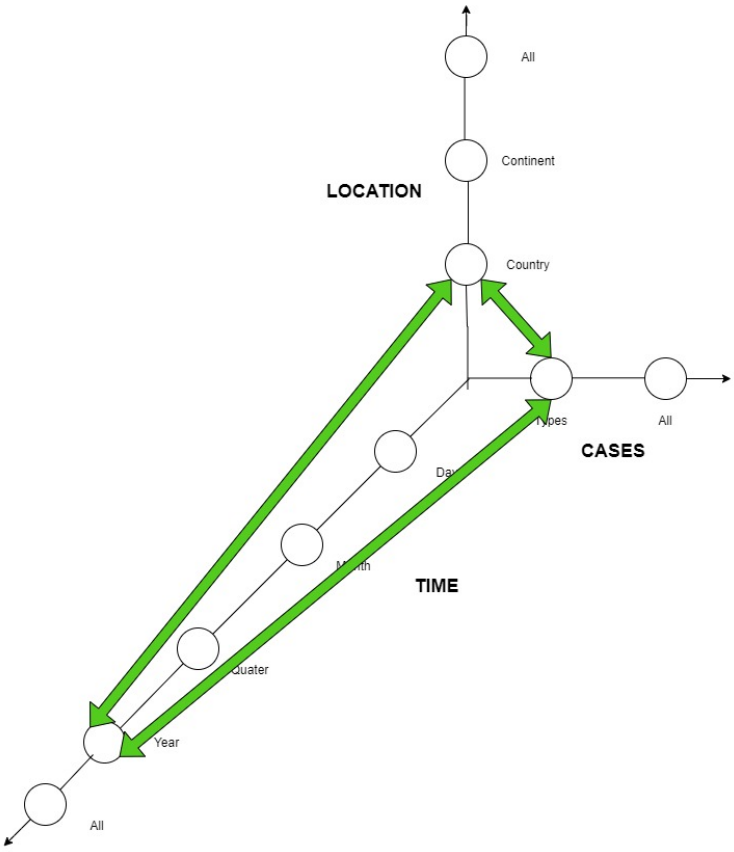

All
Continent
LOCATION
Country
Types
All
CASES
Day
Month
TIME
Quater
Year
All

Figure 2

1.b

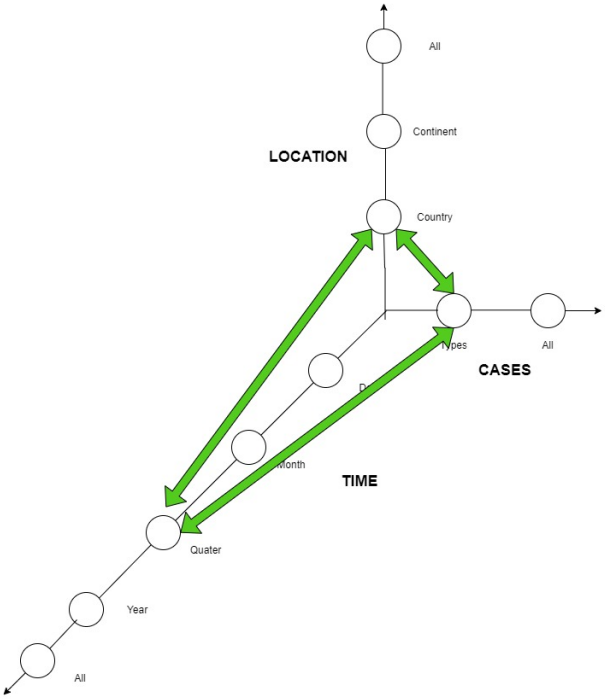


Figure 3

1.c

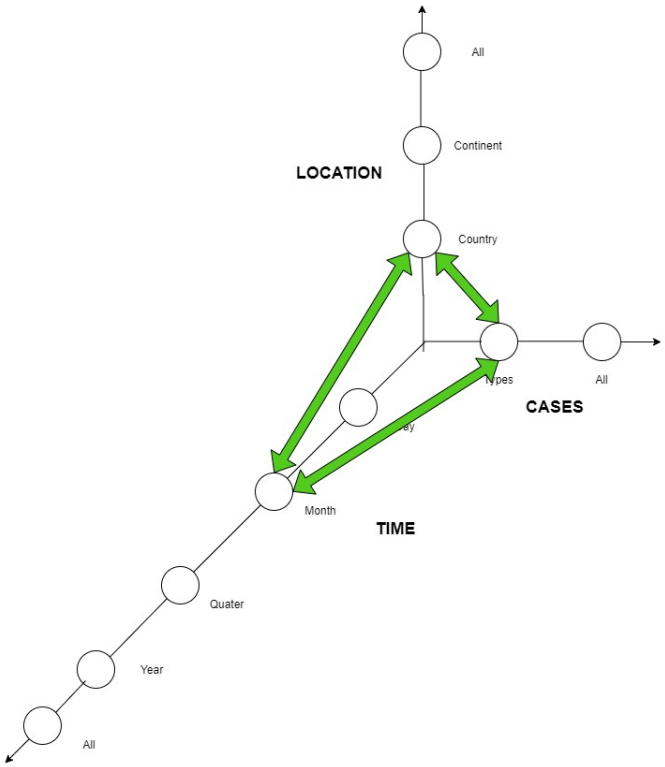


Figure 4

2.a

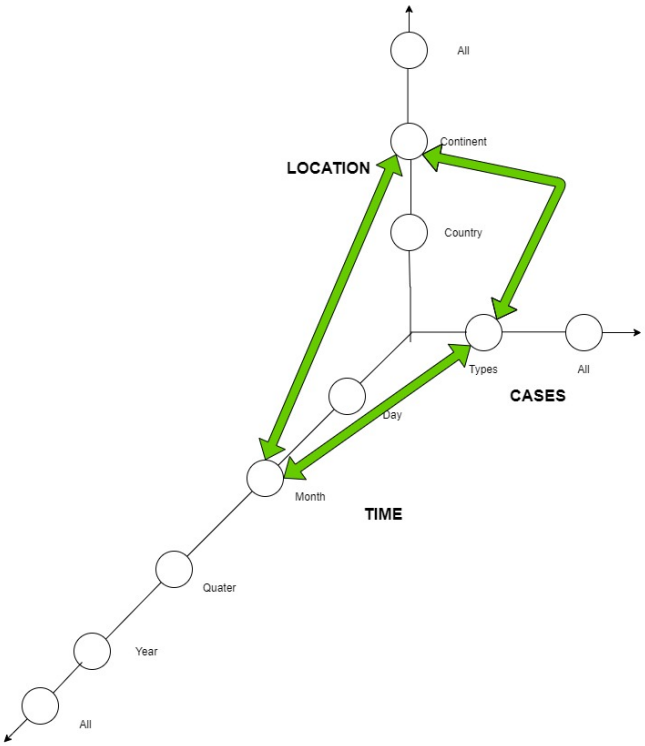

All
Continent
LOCATION
Country
Types
All
CASES
Day
Month
TIME
Quater
Year
All

Figure 5

2.b

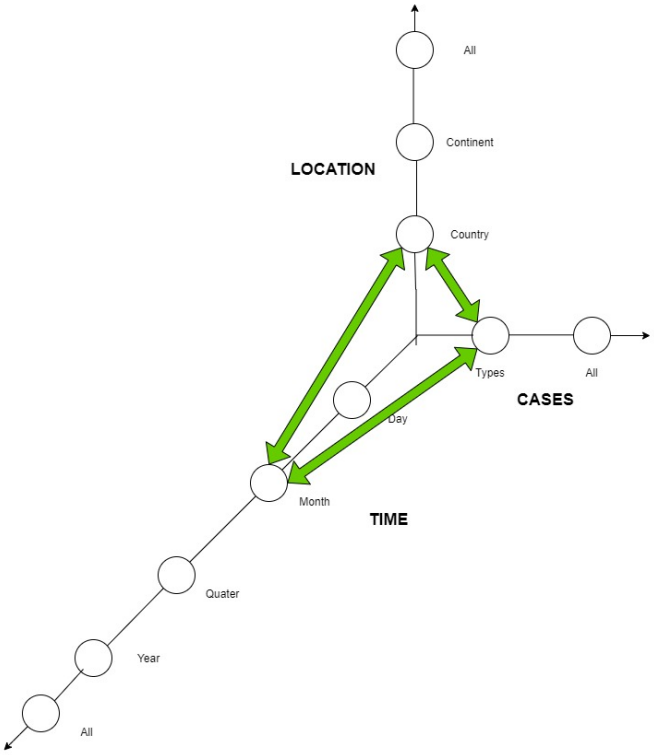


Figure 6

3.a

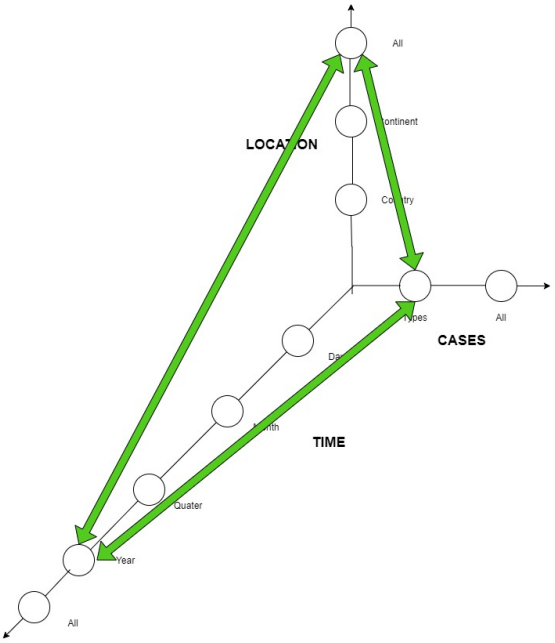


Figure 7

3.b

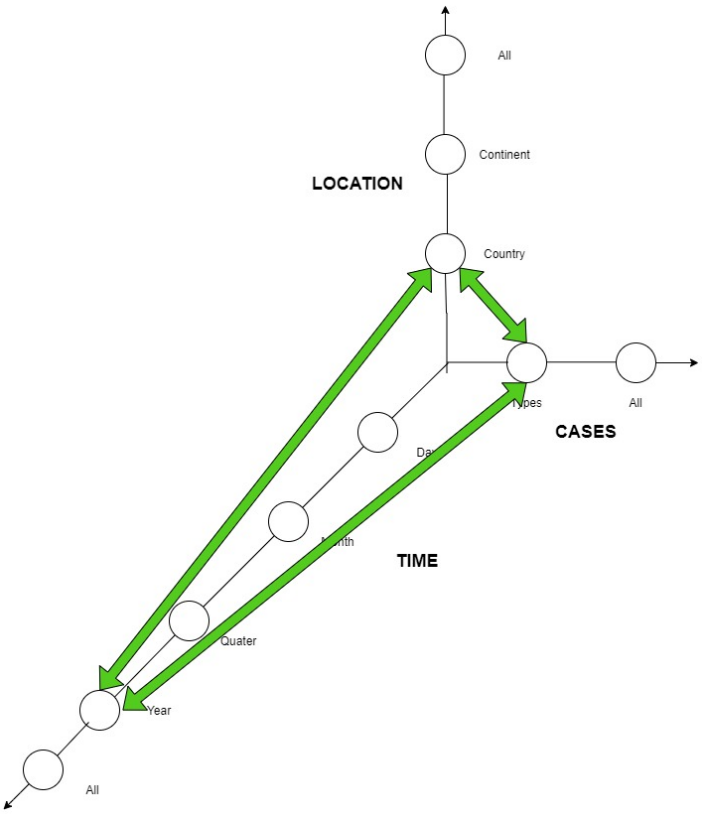

All
Continent
LOCATION
Country
Types
All
CASES
Da
Month
TIME
Quater
Year
All

Figure 8

4

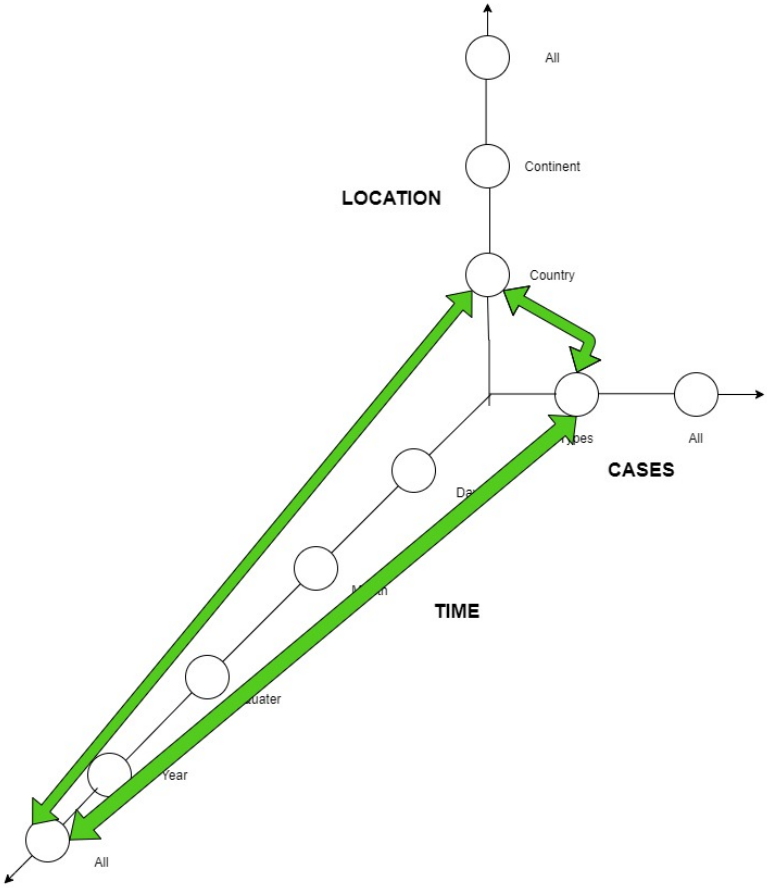


Figure 9

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.

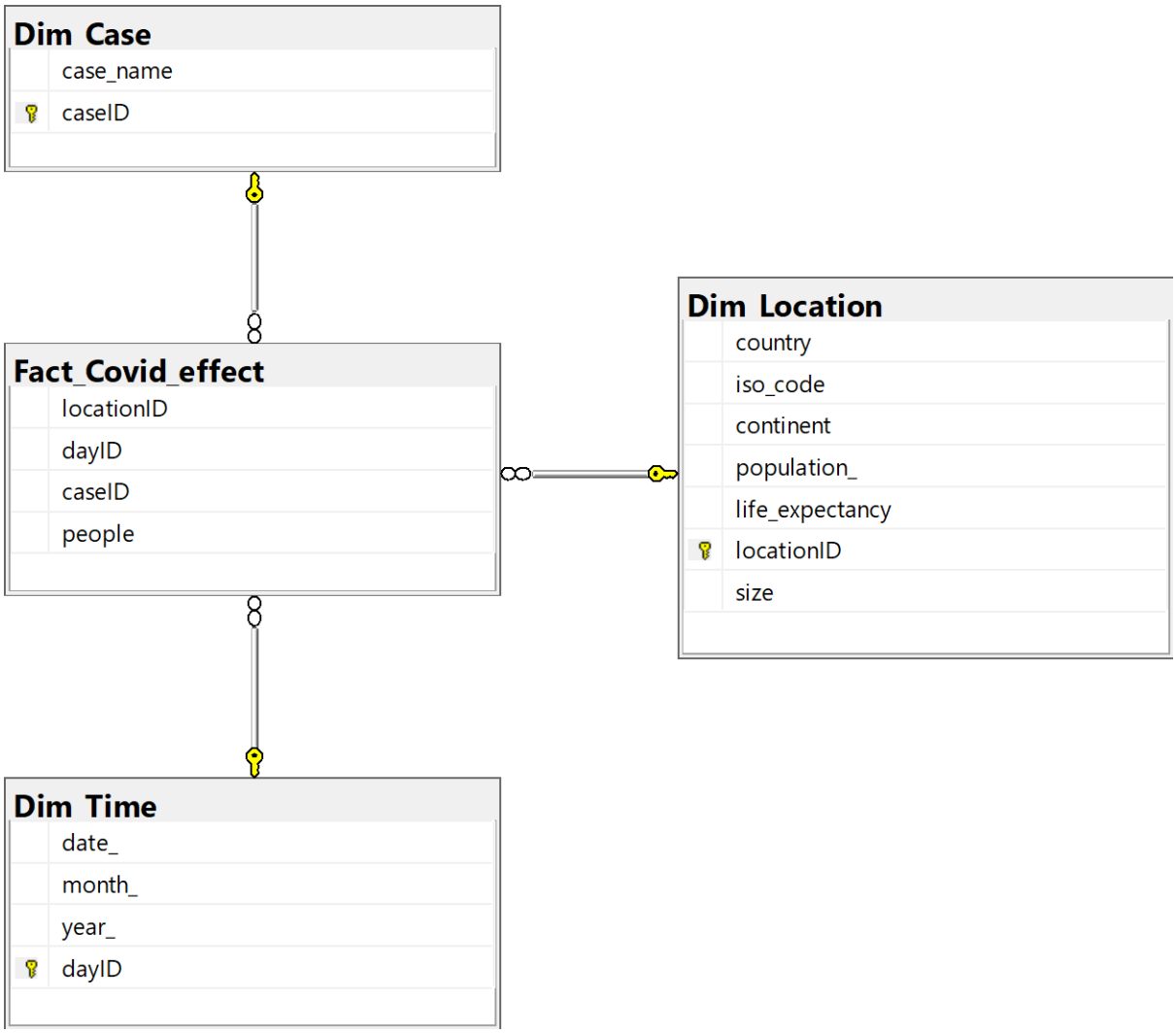


Figure 10- Star Schema

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.

1.4.1Dimension 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=Fa
```

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 ‘data_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 ‘data_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 corrsponding to each date, corressponsing 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 forgein 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’.

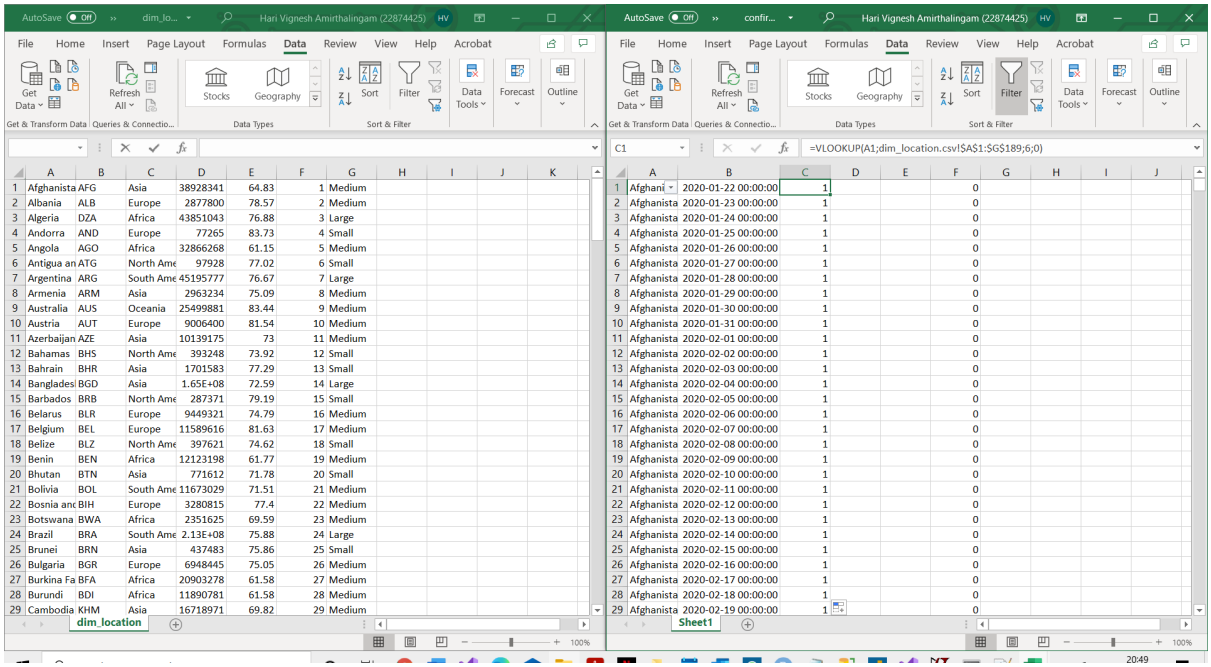


Figure 15-V-look

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 corrsponding to each date, corressponsing 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_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 forgein 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 forgein 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’.

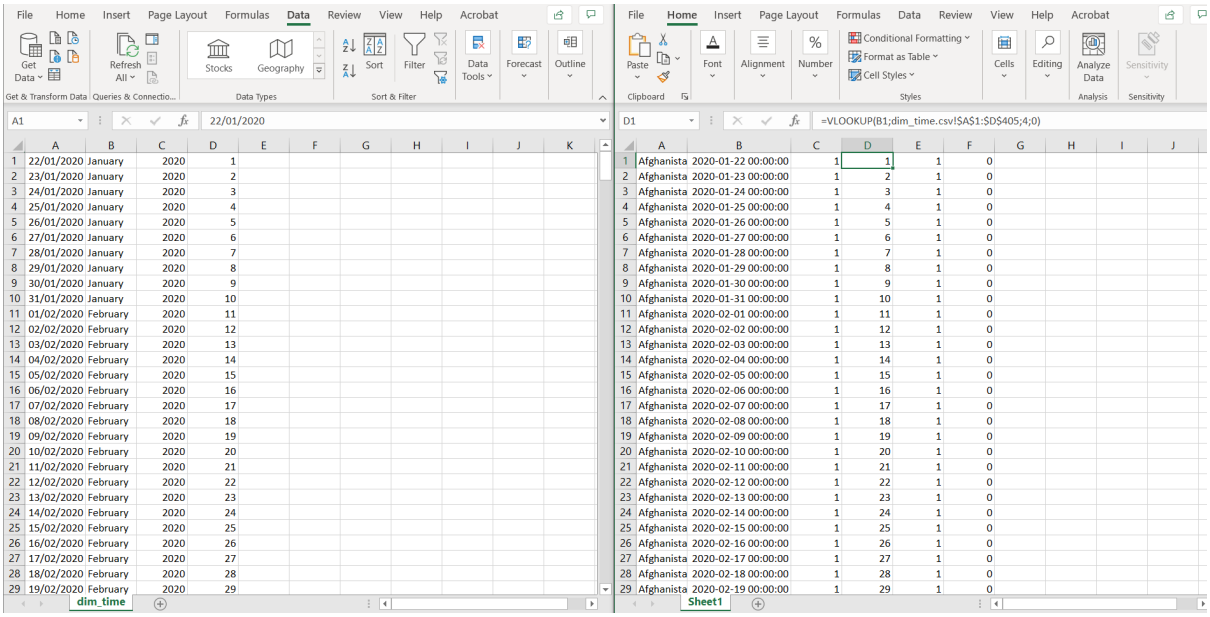


Figure 17-V-lookup

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 corrsponding to each date, corressponsing 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 forgein 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’.

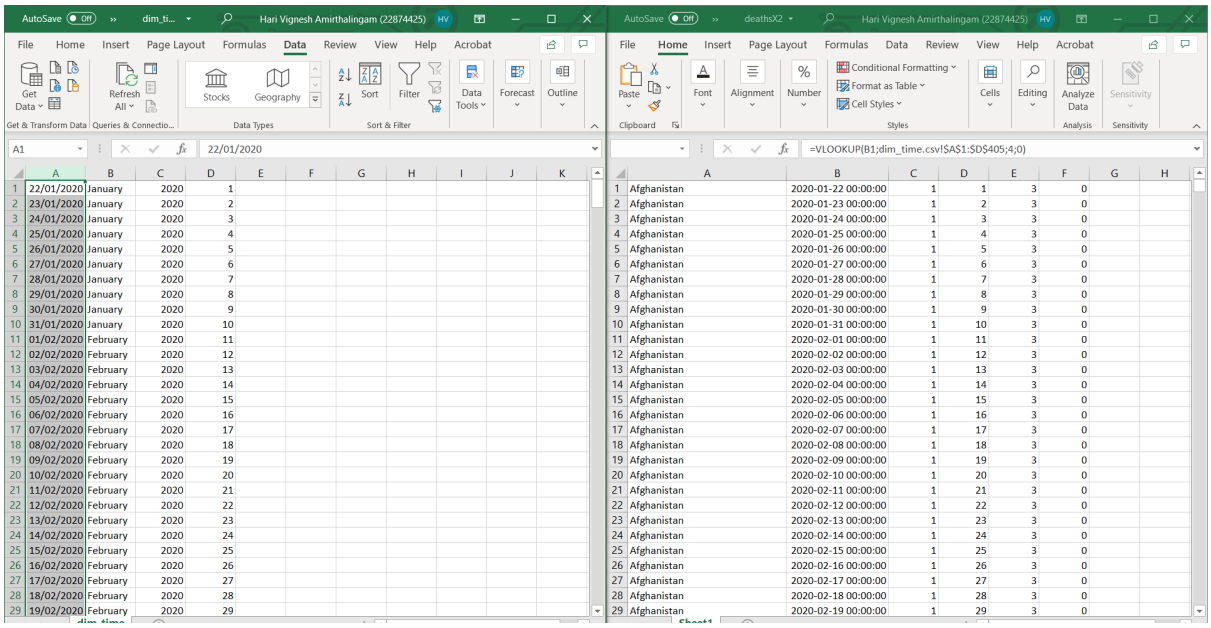


Figure 19-V-look-up

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 forgeing 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:

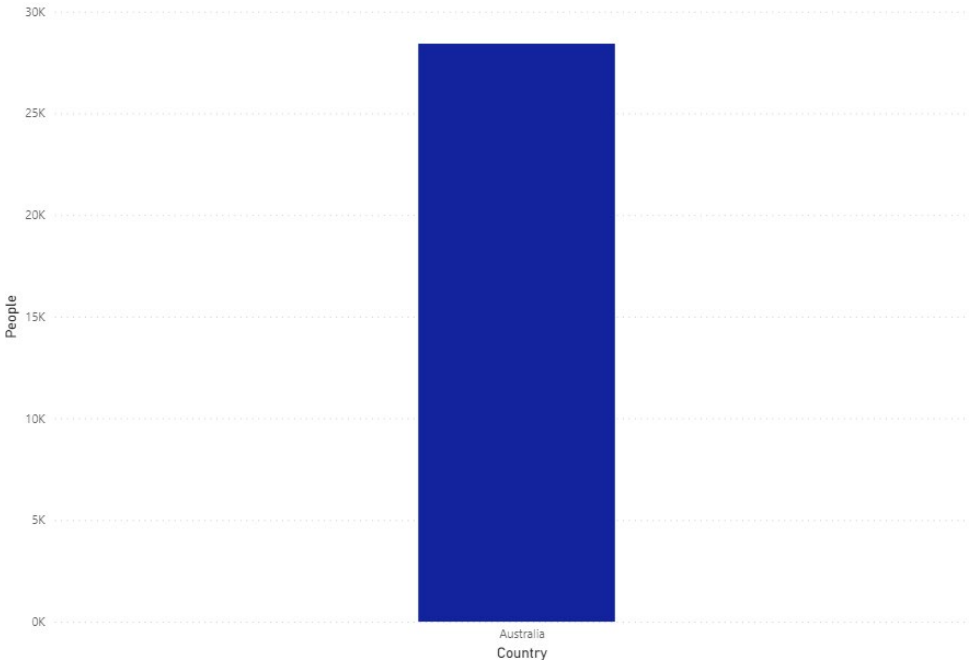


Figure 26

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

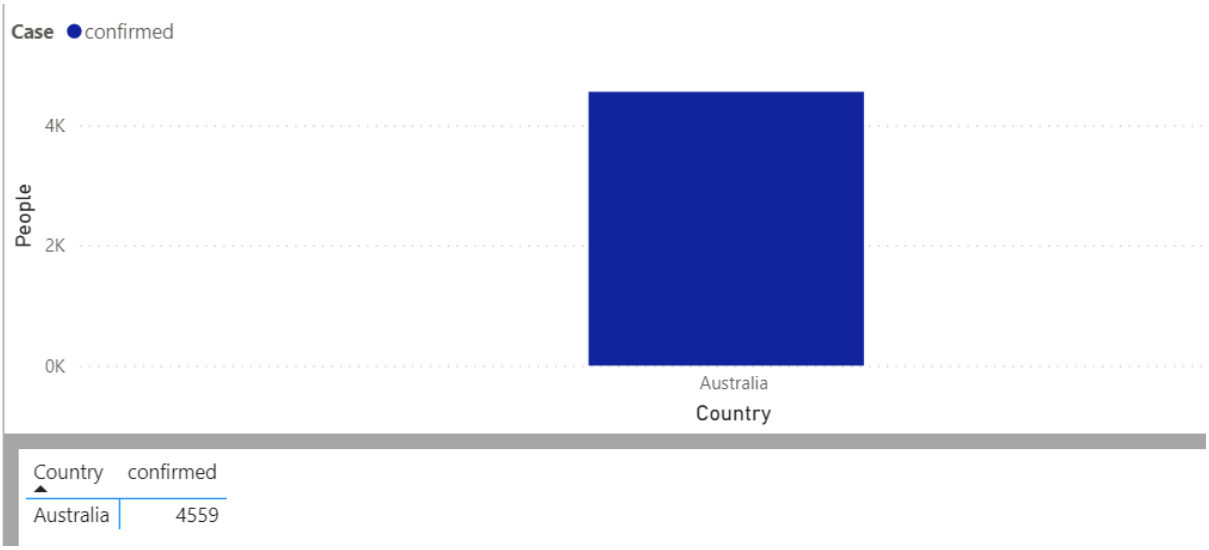


Figure 27

Quarter2:Apr-Jun

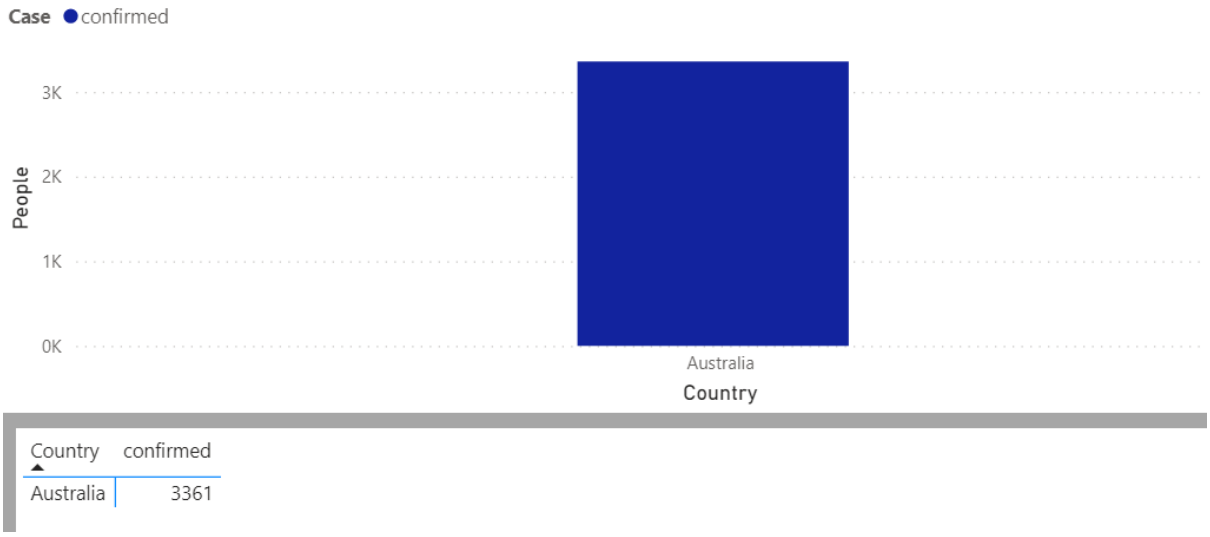


Figure 28

Quarter3: Jul-Sept

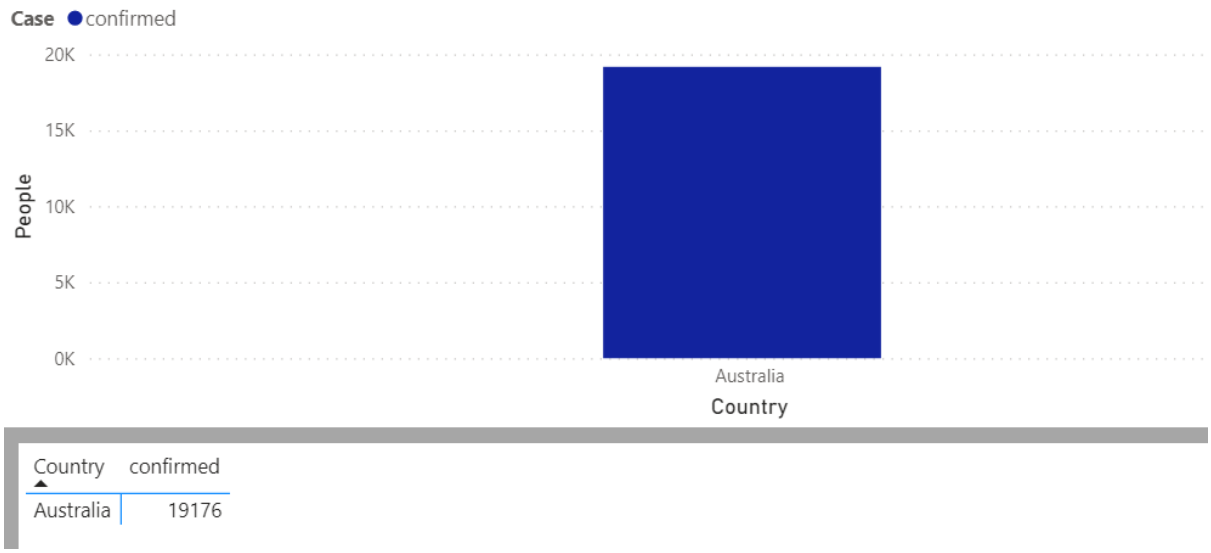


Figure 29

Quarter4: Oct-Dec

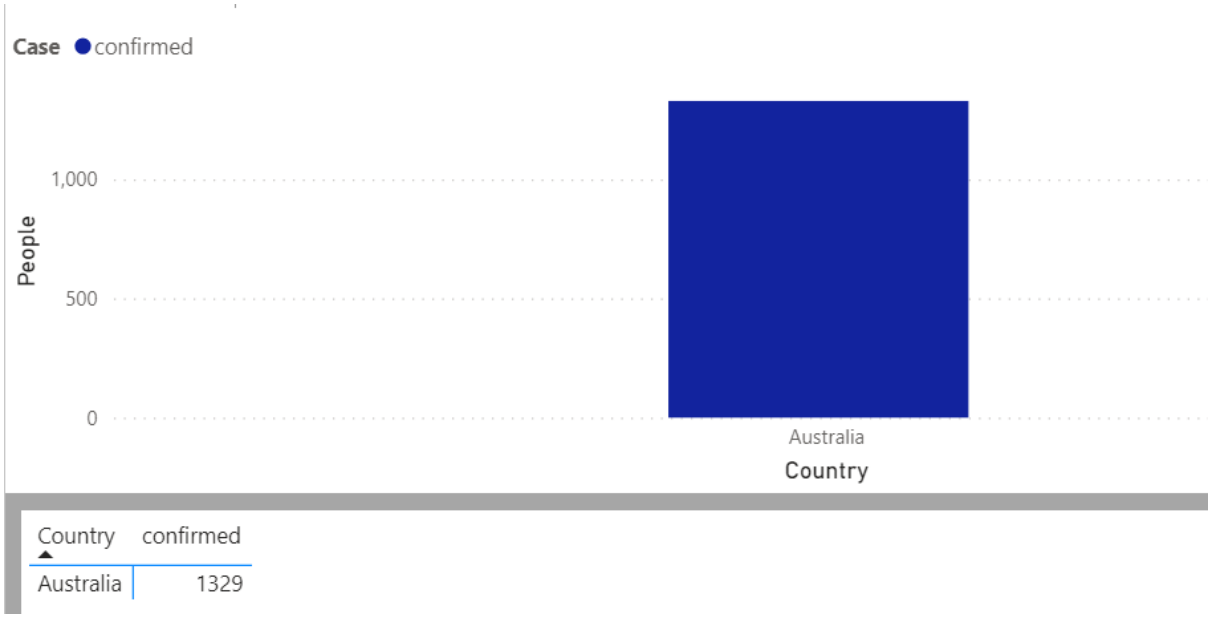
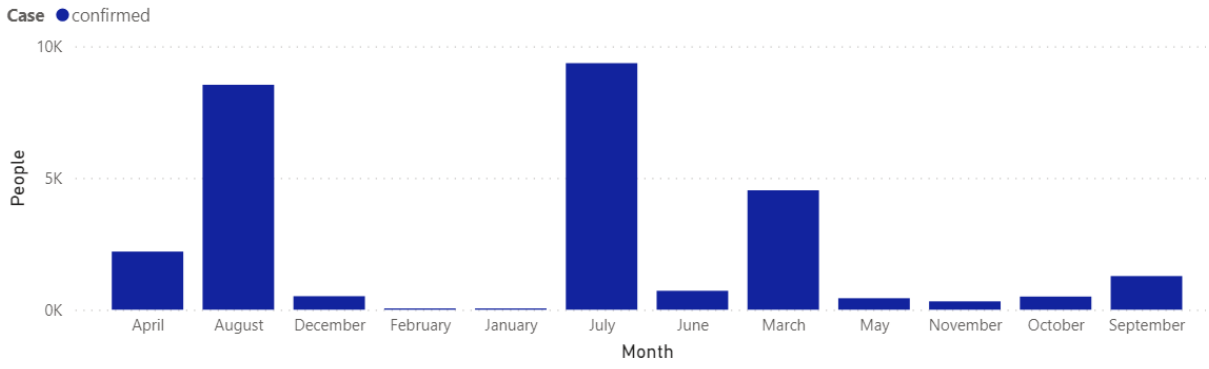


Figure 30

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.



Country, Month	confirmed
Australia, April	2207
Australia, August	8539
Australia, December	513
Australia, February	16
Australia, January	9
Australia, July	9360
Australia, June	718
Australia, March	4534
Australia, May	436
Australia, November	317
Australia, October	499
Australia, September	1277

Figure 31

2. Queries:
    a. 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

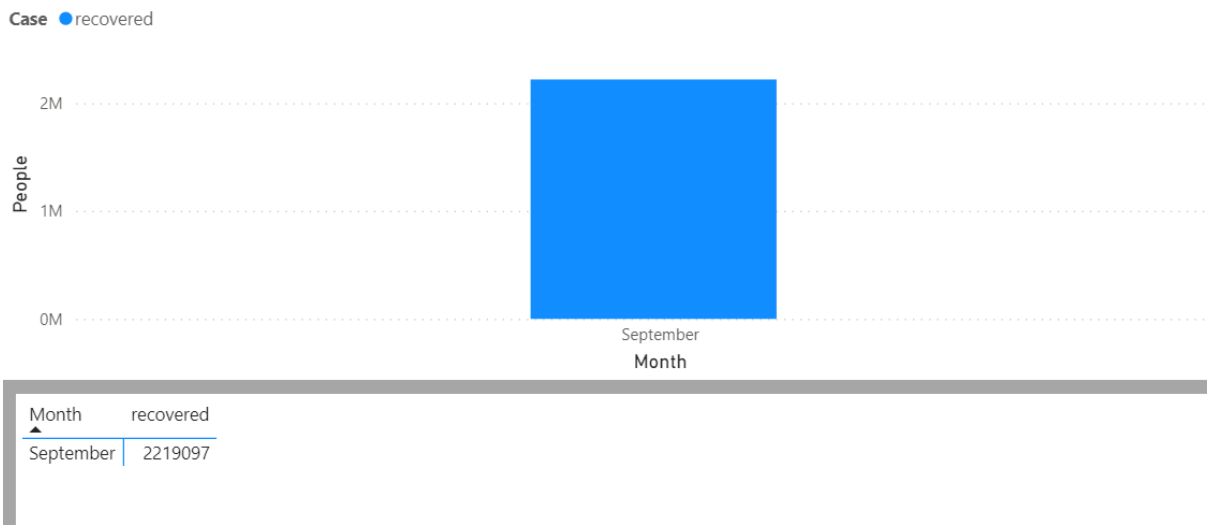


Figure 32

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

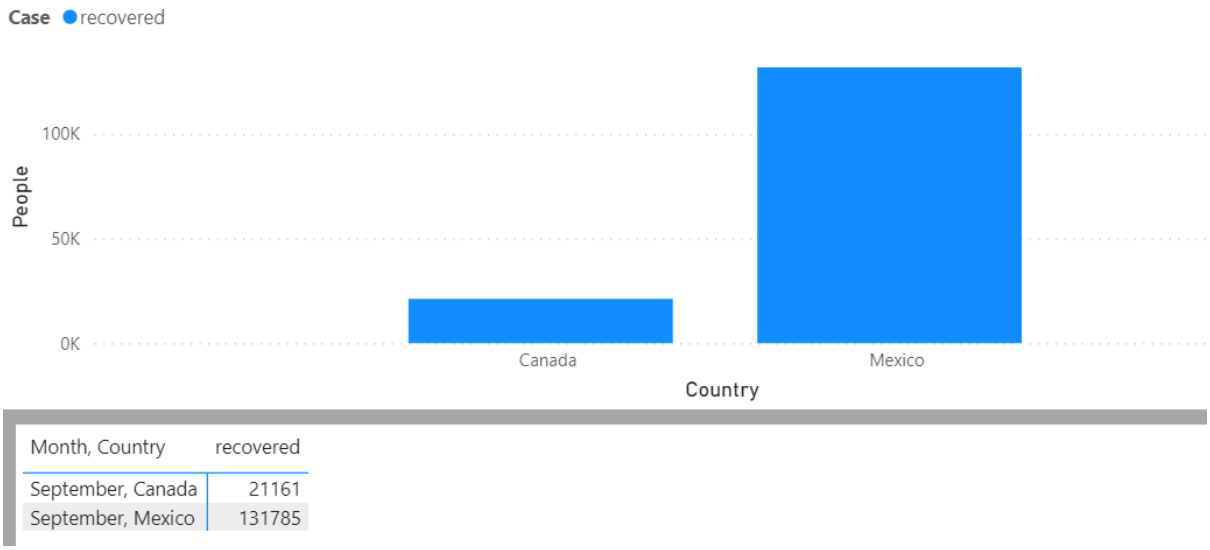


Figure 33

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

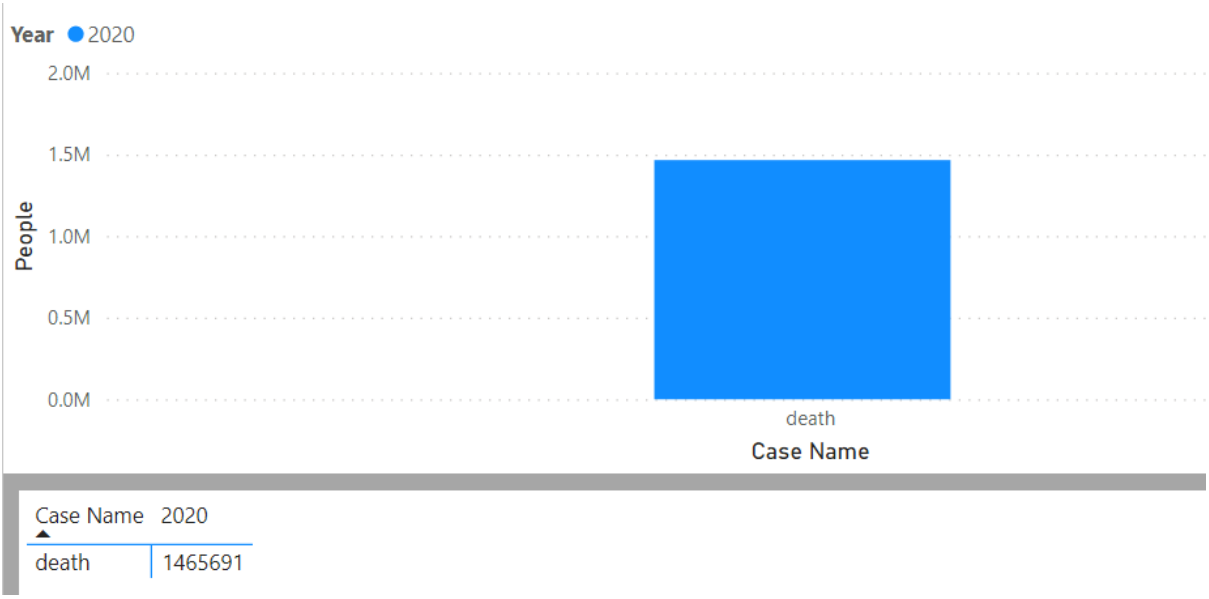
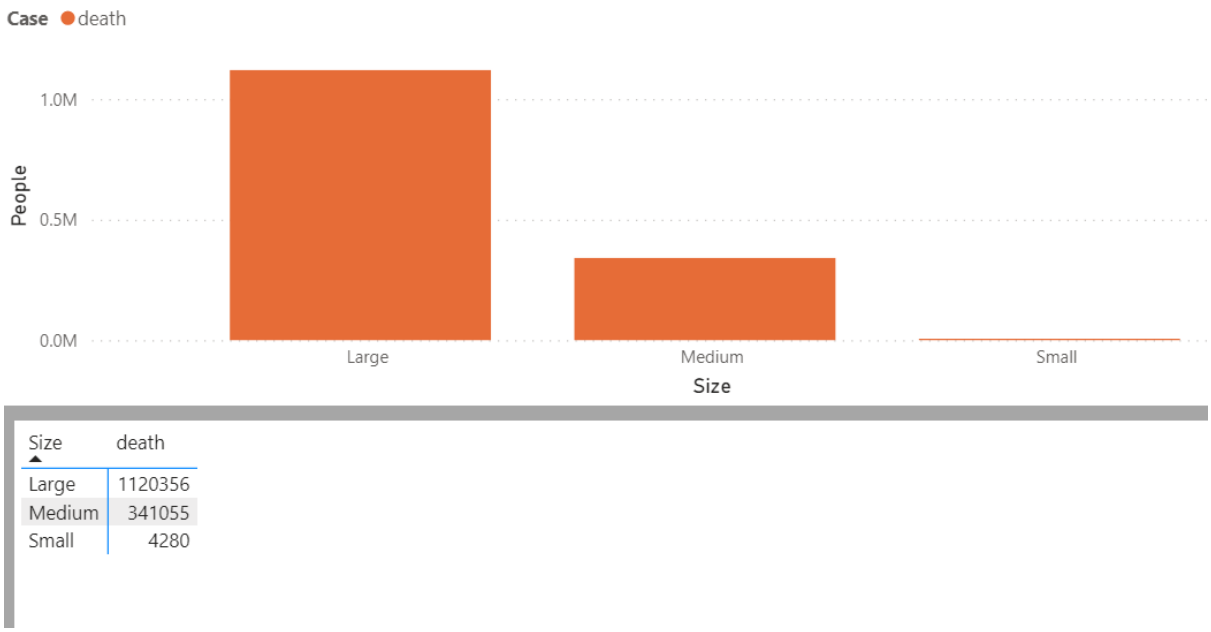


Figure 34

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



Size	death
Large	1120356
Medium	341055
Small	4280

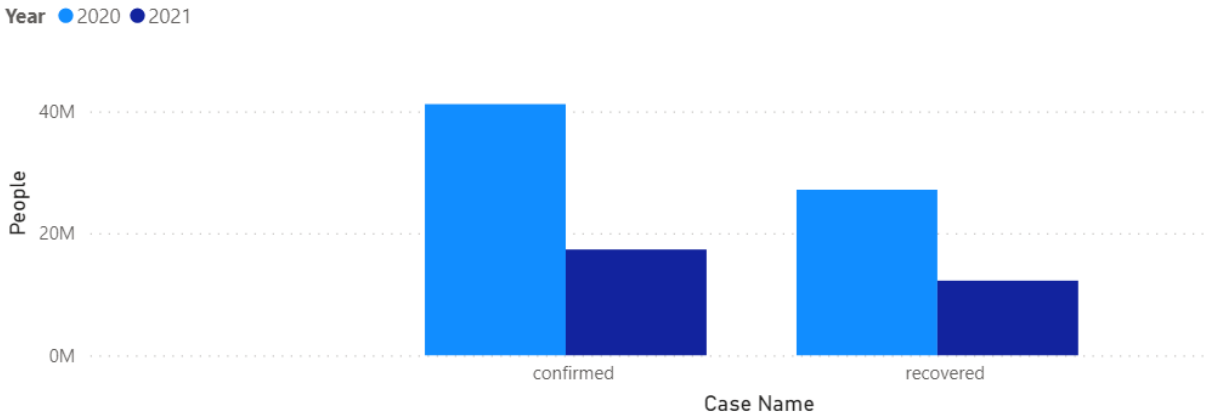
Figure 35

4. 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:



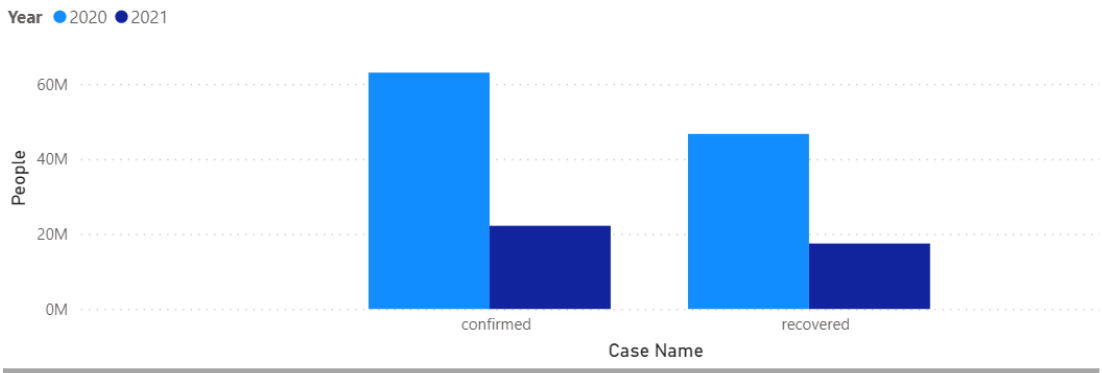
Case Name	2020	2021
confirmed	41172045	17345599
recovered	27141345	12237764

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:



Case Name	2020	2021
confirmed	63047012	22194002
recovered	46691574	17460144

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.