1. Upload COVID-19.csv to a DBMS (attached).

Solution:-Uploaded  
  
2- "Confirmed Cases" and "Fatalities" both contain "cumulative" data. Use "Windowing /Partition" statement and transform "cumulative" numbers to "daily" numbers and call them "Confirmed Daily" and "Fatalities Daily" respectively. The following is an example of "Cumulative" and "Daily" numbers.

|  |  |
| --- | --- |
| Cumulative | Daily |
| 1 | 1 |
| 10 | 9 |
| 128 | 118 |
| 235 | 107 |
| 1000 | 765 |

Solution:

SELECT \* , ConfirmedCases- LAG(ConfirmedCases,1) OVER( Partition By Country\_Region,Province\_State ORDER BY Date) AS "Confirmed Daily" ,Fatalities-LAG(Fatalities,1) OVER( Partition By Country\_Region,Province\_State ORDER BY Date) As "Fatalities Daily" FROM dbo.COVID

3- Aggregate  "ConfirmedDaily" and "FatalitiesDaily"  by "Country\_Region and WeekOfYear(Date)" and copy the aggregated data into a new table called "COVID\_19\_aggr".

|  |  |
| --- | --- |
| COVID\_19\_aggr | |
| Country\_Region | dimension |
| WeekOfYear | dimension |
| ConfirmedDaily | measurement |
| FatalitiesDaily | measurement |

Solution:

select T.Country\_Region , T.WeekOfYear , sum(T."Fatalities Daily") as "FatalitiesDaily" ,sum(T."Confirmed Daily") as "ConfirmedDaily" into COVID\_19\_aggr from (SELECT Country\_Region ,DATEPART(ww,Date) As WeekOfYear , ConfirmedCases- LAG(ConfirmedCases,1) OVER( Partition By Country\_Region,Province\_State ORDER BY Date) AS "Confirmed Daily" , Fatalities-LAG(Fatalities,1) OVER( Partition By Country\_Region,Province\_State ORDER BY Date) As "Fatalities Daily" FROM dbo.COVID) as T group by T.Country\_Region, T.WeekOfYear order by T.Country\_Region, T.WeekOfYear

4- Use "GROUP BY CUBE", "GROUP BY ROLLUP", and "GROUPING SETS" against "COVID\_19\_aggr" table.

Solution:

Solution:

"GROUP BY ROLLUP" Usage:

Explanation:

|  |  |
| --- | --- |
| 2  3  4  5  6  7  8  9 | SELECT  d1,  d2,  d3,  aggregate\_function(c4)  FROM  table\_name  GROUP BY  ROLLUP (d1, d2, d3); |

d1, d2, and d3 are the dimension columns. The statement will calculate the aggregation of values in the column c4 based on the hierarchy d1 > d2 > d3.

Query Created:

select Country\_Region, WeekOfYear, SUM(ConfirmedDaily) as ConfirmedDaily , SUM(FatalitiesDaily) as FatalitiesDaily from COVID\_19\_aggr group by rollup(Country\_Region,WeekOfYear)

**Result we get :**

**Country , week , total(fatalities),total(Confirmed Cases) ------ Country,Week wise Totals**

**Country, NULL ,Total(fatalities), Total(Confirmed Cases)------- Country Wise Totals**

**Null, NULL, Total(fatalities), Total(Confirmed Cases)-------Total of all countries in all weeks**

"GROUP BY CUBE" Usage:

SELECT

d1,

d2,

aggregate\_function(c4)

FROM

table\_name

GROUP BY

CUBE (d1, d2);

Query Created:

select Country\_Region, WeekOfYear, SUM(ConfirmedDaily) as ConfirmedDaily , SUM(FatalitiesDaily) as FatalitiesDaily from COVID\_19\_aggr group by cube (Country\_Region,WeekOfYear)

**Result we get :**

**Country , week , total(fatalities),total(Confirmed Cases) ------ Country,Week wise Totals**

**Country, NULL ,Total(fatalities), Total(Confirmed Cases)------- Country Wise Totals**

**NULL, week , total(fatalities),total(Confirmed Cases)-------- Week Wise Total confirmed cases and fatalities across all the countries in a particular week**

**Null, NULL, Total(fatalities), Total(Confirmed Cases)-------Total of all countries in all weeks**

**Grouping sets**

5- Use RANK(), DENSE\_RANK(), PERCENT\_RANK() and CUME\_DIST() against "COVID\_19\_aggr" table.

Solution:

Rank()

Query Created :

select WeekOfYear , Country\_Region, ConfirmedDaily , Rank() Over(partition by WeekOfYear order by ConfirmedDaily desc ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear

Exaplanation:

Rank function Just gives the numbers in the increasing order from 1 onwards – to the field (here number of confirmed cases) in the window( consist of confirmed cases sorted in descending order i.e. highest will rank 1 and so on so as to rank( 1 onwards) from high to low ( confirmed cases) in a partition of week of the year )

It gives same number to the field having the same value but to the any next different value give the rank of count( all previous values)+1

Similarly , we can create query for the Ranking in terms of number of fatalities ( highest country ranked 1 and so on)

Query Created :

select WeekOfYear , Country\_Region, FatalitiesDaily , Rank() Over(partition by WeekOfYear order by FatalitiesDaily desc ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear

DENSE\_RANK():

In Dense Rank , the difference from Rank() is that It gives same number to the field having the same value but to the any next different value give the rank – previous highest rank+1.

select WeekOfYear , Country\_Region, FatalitiesDaily , Dense\_rank() Over(partition by WeekOfYear order by FatalitiesDaily desc ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear

select WeekOfYear , Country\_Region, ConfirmedDaily , Dense\_Rank() Over(partition by WeekOfYear order by ConfirmedDaily desc ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear

PERCENT\_RANK():

General Usage:

PERCENT\_RANK() OVER (

    [PARTITION BY partition\_expression, ... ]

    ORDER BY sort\_expression [ASC | DESC], ...

)

PARTITION BY

The PARTITION BY clause distributes the rows into multiple partitions to which the PERCENT\_RANK() function is applied. The PARTITION BY clause is optional. If you skip it, the function will treat the whole result set as a single partition.

### ORDER BY

The ORDER BY clause specifies the logic order of rows in each partition. Because PERCENT\_RANK() is order sensitive, the order\_by\_clause is required.

Return value

The result of PERCENT\_RANK() is greater than 0 and less than or equal to 1.

0 < PERCENT\_RANK() <= 1

So , in accordance our above queries can be modified as :

select WeekOfYear , Country\_Region, ConfirmedDaily , Format(Percent\_Rank() Over(partition by WeekOfYear order by ConfirmedDaily desc ),'P' ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear

Here it assigns the 0 percent to the country with the highest number of cases in a particular week showing that 0 percent rows are ahead of it.

**Same values get same percentage.**

**And the next different value get : count(values before)/(total\_count in the window -1) \* 100 as Percent\_Rank() in our current formatting**

select WeekOfYear , Country\_Region, ConfirmedDaily , Format(Percent\_Rank() Over(partition by WeekOfYear order by ConfirmedDaily ),'P' ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear , ConfirmedDaily desc

Here it assigns the 100 percent to the country with the highest number of cases in a particular week showing that 0 percent rows are ahead of it.

So country with 100 percent is the highest cases one and 0 percent is the lowest one just opposite of the above.

To realise the difference between Cum\_Dist and Percent\_rank:

|  |  |
| --- | --- |
| Department LastName Rate CumeDist PctRank | |
| ---------------------- ---------------------- --------------------- ---------------------- ---------------------- | | |
| Document Control Arifin 17.7885 1 1 |
| Document Control Norred 16.8269 0.8 0.5 |
| Document Control Kharatishvili 16.8269 0.8 0.5 |
| Document Control Chai 10.25 0.4 0 |
| Document Control Berge 10.25 0.4 0 |
| Information Services Trenary 50.4808 1 1 |
| Information Services Conroy 39.6635 0.9 0.888888888888889 | | |
| Information Services Ajenstat 38.4615 0.8 0.666666666666667 | | |
| Information Services Wilson 38.4615 0.8 0.666666666666667 | | |
| Information Services Sharma 32.4519 0.6 0.444444444444444 | | |
| Information Services Connelly 32.4519 0.6 0.444444444444444 | | |
| Information Services Berg 27.4038 0.4 0 |
| Information Services Meyyappan 27.4038 0.4 0 |
| Information Services Bacon 27.4038 0.4 0 |
| Information Services Bueno 27.4038 0.4 0 |

CUME\_DIST():

Corresponding queries to Percent\_Rank()

select WeekOfYear , Country\_Region, ConfirmedDaily , Cume\_dist() Over(partition by WeekOfYear order by ConfirmedDaily desc ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear

select WeekOfYear , Country\_Region, ConfirmedDaily ,Cume\_dist() Over(partition by WeekOfYear order by ConfirmedDaily ) as Country\_Rank from COVID\_19\_aggr order by WeekOfYear , ConfirmedDaily desc

Cume distance just put all equal values in one group so we have

G1, G2 , G3……………………… so on groups

So it assigns the values 0 to 1 to the values in the window in any order whether ascending or descending , starting with the lowest (0< x< 1) and this value of x is ( considering G1, G2 , G3……………………… so on groups as order of the groups)

For any group N : (Num of all values in the Gn + all values before)/total number of values in the window

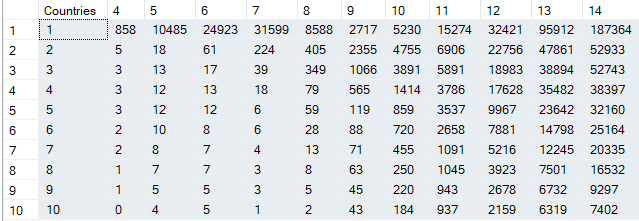
6- Build two PIVOT tables as follows.  
  
PIVOT #1:  
Dimension 1: WeekOfYear   
Dimension 2: Top 10 Country\_Region  
Measurement 1: ConfirmedDaily  
  
PIVOT #2:  
Dimension 1: WeekOfYear   
Dimension 2: Top 10 Country\_Region  
Measurement 2: FatalitiesDaily

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Week4 | Week5 | Week6 | Week7 | Week8 | Week9 | Week10 | Week11 | Week12 | Week13 | Week14 |
| Country1 | ConfirmedDaily and FatalitiesDaily | | | | | | | | | | |
| Country2 |
| Country3 |
| Country4 |
| Country5 |
| Country6 |
| Country7 |
| Country8 |
| Country9 |
| Country10 |

Queries:

Pivot Table 1:

select \* from (select WeekOfYear,ConfirmedDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( sum(ConfirmedDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table



Query for just selecting top 10 countries per week :

select \* from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10

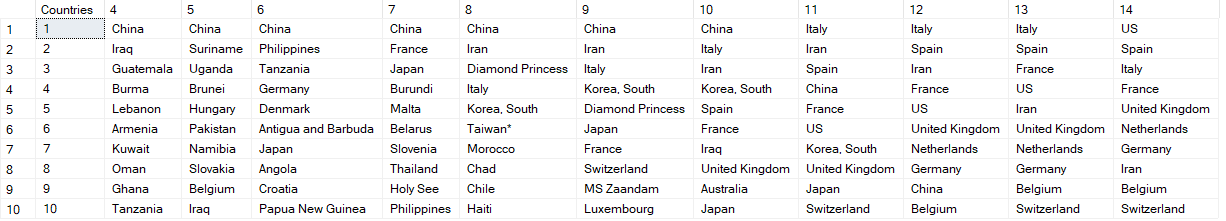
Pivot 1 with countries name:

select \* from (select WeekOfYear,Country\_Region , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( max(Country\_Region) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table

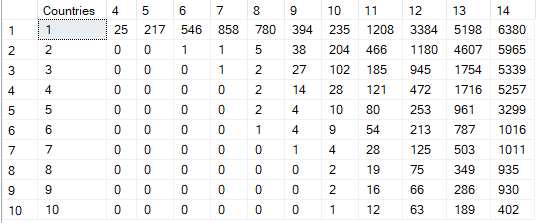


Pivot 2:

select \* from (select WeekOfYear,Country\_Region , T.row as "Countries" from (select WeekOfYear,Country\_Region , FatalitiesDaily , row\_number() over( partition by WeekOfYear order by FatalitiesDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( max(Country\_Region) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table



select \* from (select WeekOfYear,FatalitiesDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , FatalitiesDaily , row\_number() over( partition by WeekOfYear order by FatalitiesDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( max(FatalitiesDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table



**7- Use DRILL DOWN, DRILL UP, SLICING and DICING against the above PIVOT tables**.

### ****Roll up (drill-up):****

ROLLUP is used in tasks involving subtotals. It creates subtotals at any level of aggregation needed, from the most detailed up to a grand total i.e. climbing up a concept hierarchy for the dimension such as time or geography. **Example :** A Query could involve a ROLLUP of year>month>day or country>state>city. When a roll-up is performed, one or more dimensions from the data cube are removed because the output would display blank for certain rows.

### ****Drill down (Roll down)****

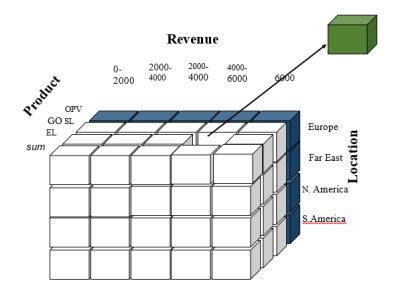
This is a reverse of the ROLL UP operation discussed above. The data is aggregated from a higher level summary to a lower level summary/detailed data.

### ****Slicing :****

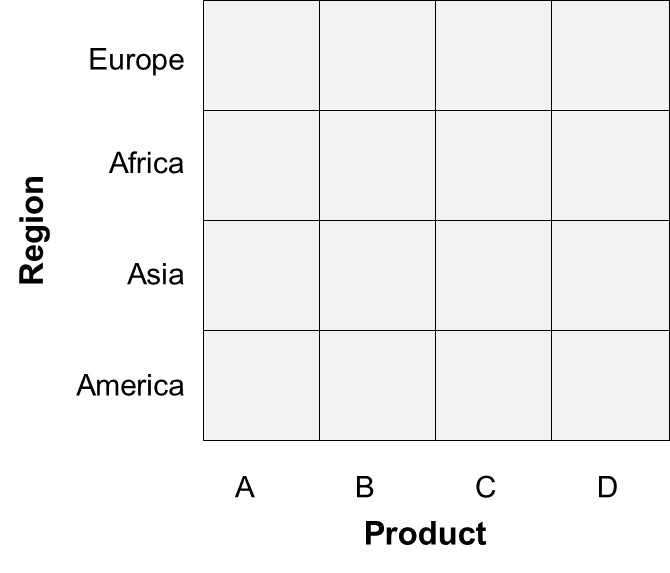
A slice in a multidimensional array is a column of data corresponding to a single value for one or more members of the dimension. It helps the user to visualize and gather the information specific to a dimension. When you think of slicing, think of it as a specialized filter for a particular value in a dimension. For instance, if a user wanted to know the total number of OPV products sold across all of the market locations (Europe, Far-Ease, North America, South America,) the user would perform a horizontal slice

### ****Dicing:****

Dicing is similar to slicing, but it works a little bit differently. When one thinks of slicing, filtering is done to focus on a particular attribute. Dicing, on the other hand, is more of a zoom feature that selects a subset over all the dimensions, but for specific values of the dimension.



In our case of Pivot Table we have two dimensions of Data :



One axis is the Weeks from 4 to 14 and other is the Countries( Top 10 countries)

1. For Drill Up we can aggregate Data Over dimensions :

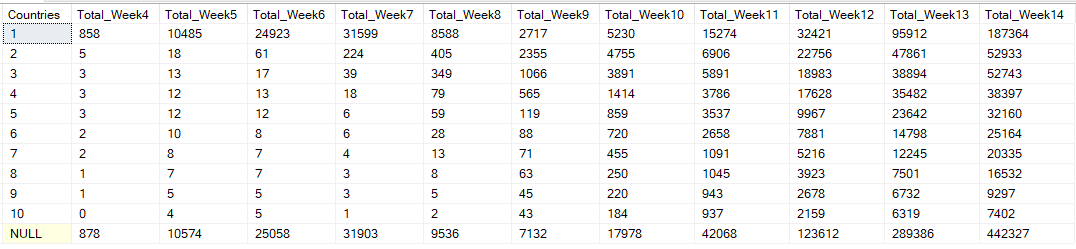
**Aggregating data of top 10 countries for all weeks :**

select sum(piv.[4]) as Total\_Week4 ,sum(piv.[5]) as Total\_Week5,sum(piv.[6]) as Total\_Week6 ,sum(piv.[7]) as Total\_Week7 ,sum(piv.[8]) as Total\_Week8 ,sum(piv.[9]) as Total\_Week9 ,sum(piv.[10]) as Total\_Week10 ,sum(piv.[11]) as Total\_Week11 ,sum(piv.[12]) as Total\_Week12 ,sum(piv.[13]) as Total\_Week13 ,sum(piv.[14]) as Total\_Week14 from (select \* from (select WeekOfYear,ConfirmedDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( sum(ConfirmedDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table) as piv



Or we can use the Group by roll up where we are aggregating the one dimension :

select sum(piv.[4]) as Total\_Week4 ,sum(piv.[5]) as Total\_Week5,sum(piv.[6]) as Total\_Week6 ,sum(piv.[7]) as Total\_Week7 ,sum(piv.[8]) as Total\_Week8 ,sum(piv.[9]) as Total\_Week9 ,sum(piv.[10]) as Total\_Week10 ,sum(piv.[11]) as Total\_Week11 ,sum(piv.[12]) as Total\_Week12 ,sum(piv.[13]) as Total\_Week13 ,sum(piv.[14]) as Total\_Week14 from (select \* from (select WeekOfYear,ConfirmedDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( sum(ConfirmedDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table) as piv



So here we are aggregating across the country dimension using group roll up function.

1. Slicing

In order to do slicing across particular dimension. Like :

1. selecting particular week details for top 10 countries (on week dimension)
2. 5th ranked country details across all the weeks( on country dimension)

For this we have created a stored procedure :

Create procedure slicing

(@dim bit, @number int)

as

if @dim =1

begin

select \* from (select \* from (select WeekOfYear,ConfirmedDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( sum(ConfirmedDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table) as piv where Countries = @number

end

else

begin

declare @temp int

set @temp=@number+3

select Case @temp When 4 then [4] When 5 then [5] When 6 then [6] When 7 then [7] When 8 then [8] When 9 then [9] When 10 then [10] When 11 then [11] When 12 then [12] When 13 then [13] When 14 then [14] else NULL End as "Week" from (select \* from (select WeekOfYear,ConfirmedDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( sum(ConfirmedDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table) as piv

end

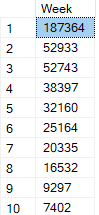
Go

For execution we do :

**Dim** is a bit parameter for selecting the dimension . In our case , dim = 0 is the slicing for a particular week and dim=1 is the slicing across the particular country

**Number** parameter is for selecting the particular number

exec dbo.slicing @dim=0,@number=11



exec dbo.slicing @dim=1,@number=7



1. Dicing

Dicing is just ahead of slicing where we were cutting across one dimension but here we are cutting across various dimension to get the result. **Dicing** corresponds to a range selection on one or more dimensions

In our case , we can look the country dimension as X and the week dimension as Y and F(X,Y) returning the value at particular point(X,Y) is a dicing operation.

In order to implement this , we have created the stored procedure to pass on the X, Y and get the result:

Create procedure dicing

(@week bit, @country int)

as

declare @temp int

set @temp=@week+3

select Case @temp When 4 then [4] When 5 then [5] When 6 then [6] When 7 then [7] When 8 then [8] When 9 then [9] When 10 then [10] When 11 then [11] When 12 then [12] When 13 then [13] When 14 then [14] else NULL End as "Week" from (select \* from (select WeekOfYear,ConfirmedDaily , T.row as "Countries" from (select WeekOfYear,Country\_Region , ConfirmedDaily , row\_number() over( partition by WeekOfYear order by ConfirmedDaily desc ) as row from COVID\_19\_aggr) as T where T.row <=10 ) as m Pivot ( sum(ConfirmedDaily) FOR m.WeekOfYear IN ( [4], [5], [6],[7],[8],[9],[10],[11],[12],[13],[14] ) ) AS pivot\_table) as piv where Countries=@country

Go

Execution:

exec dbo.dicing @week=1,@country=5



Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/queries/from-using-pivot-and-unpivot?view=sql-server-ver15>

https://www.sqlservertutorial.net/sql-server-basics/

<https://www.sqlshack.com/dynamic-pivot-tables-in-sql-server/>

<https://blogs.perficient.com/2017/08/02/data-cube-operations-sql-queries/>