**Project : Acquisition**

**Optiveriance is planning to be merged with PDCL Ltd. Brown who is the CEO of PDCL is looking forward to a smooth transition and so after multiple rounds of discussion. They have decided to do audit of the business Infrastructure of Optiveriance.**

**Justin who the Chief Data Officer along with his team started auditing on the following grounds:-**

• What is the Total Sales.

• What about the Total Quantity.

• Profit for the said period and the margin on the same.

• Need to have a details Performance LY for any selected Year.

• Comparing the Performance vs LY.

• Sales 2yrs ago for any Selected Year.

• It is important to know the moving average in terms of Profit & Sales.

• A date table will make the analysis stable.

• Quarter wise analysis is also important, may be slicer.

• Total Sales, Profit and % Profit can be shown as combo with card and line chart.

• Show the sales Comparison between Cumulative Performance vs Cumulative Performance LY using a area chart.

• Top Performer in terms of Product and Customer top 7 will give a clear idea.

• Performance in terms of region sales needs to workout and also needs to work on as what will be the best way to visualise it except Bar Graph.

**We have given "Optiveriance" Excel Data. The data consists of following tables:**

 Customer Data table

 Metric Selection table

 Product Data table

 Regions table

 Sales table

First, we load the data into power bi desktop:

To do this, click on "Home" section and click on "Get data" and select "Excel data" as the data is in excel (.xlsx) format.

**Initial observation using power query editor:**

The tables look fine with proper datatypes, headers and no null values in between.

Lets consider this Sales table as fact table and remaining tables as dimension tables.

Creating a measure table to keep all the important measures at one place:

Before calculating measures, it is important to keep all the measures in a separate measure table in power bi desktop for our convenience.

So, lets create a measure table "Important Measures". to do this:

Under "**Home**" section, click on "**Enter data**" and give it a name "**Important Measures**" then click on "**Load**".

**Task 1) What is the Total Sales.**

Here we can find Total Sales in two ways:

First way using SUM on Total Revenue column:

Below DAX formula gives Total Sales using SUM function on Total Revenue column of Sales table:

**Total** Sales = SUM(Sales[Total Revenue])

Second way using SUMX on the product of Quantity and Unit Price:

Right click on**Important Measures**table under **Fields**section and click on **New measure**

Name this New measure as **Total Sales**and write below DAX formula:

Total Sales = SUMX(Sales,Sales[Order Quantity]\*Sales[Unit Price])

Here, SUMX function performs row-level evaluation of sum of product of  "Order Quantity" column and "Unit Price" column in Sales table.

**Task 2) What about the Total Quantity.**

To do this:

Right click on**Important Measures**table under **Fields**section and click on **New measure.**

Name this New measure as **Total Quantity**and write below DAX formula:

**Total** Quantity = SUM(Sales[**Order** Quantity])

Here, SUM function adds all the numbers of "Order Quantity" column of Sales table.

**Task 3) Profit for the said period and the margin on the same.**

To calculate Total Profit,

first we need to calculate**Total Cost** using below DAX formula:

Total Cost = SUMX(Sales,Sales[Order Quantity]\*Sales[Total Unit Cost])

Here, SUMX function performs row-level evaluation of sum of product of "Order Quantity" column and "Total Unit Cost" column in Sales table.

**Total Profit** can be calculated using below DAX formula:

**Total** Profit = [Total Sales]-[Total Cost]

Here, we are subtracting "Total Cost" measure from "Total Sales" measure to get Total Profit measure.

***All the above measures can be viewed using card visual:***

Graphical user interface, text, whiteboard

Description automatically generated

**Profit Margin**can be calculated using below DAX formula:

**Profit** Margin = DIVIDE([Total Profit],[Total Sales],0)

Here, By dividing "Total Profit" measure with "Total Sales", we get "Profit Margin" measure.

To convert the result Profit Margin in terms of percentage:

Select**Profit Margin**measure under **Important Measures** of **Fields**section and click on **Measure tools**on the top toolbar. Under **Measure tools**section**,** click on **%**icon.

**Now the profit margin for products can be viewed using text table visual as shown below:**

Table

Description automatically generated

**Task 4) A date table will make the analysis stable.**

We need to create a Date table to create time intelligence measures and to make the analysis stable.

Before creating Date Calendar, we need to know the range of dates i.e "Start Date" and "End Date"

Based on the dates range of **OrderDate**column, we need to decide the date range.

To know this:

Click on the down arrow icon of **OrderDate**column of **Sales**table in the Data view.

By scrolling down from top to bottom under Date filters, we can know Start Date and End Date.

Here, in this scenario:

**Start Date**is 01-01-2014 and **End Date**is 31-12-2016

**Lets create a Date table using M function code:**

**step 1)** Click on **Transform data**under **Home**section to switch to power query editor.

**step 2)**In power query editor, click on **New Source**under **Home**menu and click on **Blank Query**option. then, a default blank query with the name Query1 gets created.

Here lets rename **Query1**to **DateQuery**(renaming query is optional here but it is better to give a proper name for any query for easy identification)

**step 3)** Then, click on**Advanced Editor**under**Home**menu. a window pop ups with default piece of code.

**step 4)** Now, replace the default code with the below entire M function code:

**let** fnDateTable = (StartDate **as** date, EndDate **as** date, FYStartMonth **as** number) **as** table =>

**let**

DayCount = Duration.Days(Duration.From(EndDate - StartDate)),

Source = List.Dates(StartDate,DayCount,#duration(1,0,0,0)),

TableFromList = Table.FromList(Source, Splitter.SplitByNothing()),

ChangedType = Table.TransformColumnTypes(TableFromList,{{"Column1", **type** date}}),

RenamedColumns = Table.RenameColumns(ChangedType,{{"Column1", "Date"}}),

InsertYear = Table.AddColumn(RenamedColumns, "Year", each Date.Year([Date]),**type** text),

InsertYearNumber = Table.AddColumn(RenamedColumns, "YearNumber", each Date.Year([Date])),

InsertQuarter = Table.AddColumn(InsertYear, "QuarterOfYear", each Date.QuarterOfYear([Date])),

InsertMonth = Table.AddColumn(InsertQuarter, "MonthOfYear", each Date.Month([Date]), **type** text),

InsertDay = Table.AddColumn(InsertMonth, "DayOfMonth", each Date.Day([Date])),

InsertDayInt = Table.AddColumn(InsertDay, "DateInt", each [Year] \* 10000 + [MonthOfYear] \* 100 + [DayOfMonth]),

InsertMonthName = Table.AddColumn(InsertDayInt, "MonthName", each Date.ToText([Date], "MMMM"), **type** text),

InsertCalendarMonth = Table.AddColumn(InsertMonthName, "MonthInCalendar", each (**try**(Text.Range([MonthName],0,3)) otherwise [MonthName]) & " " & Number.ToText([Year])),

InsertCalendarQtr = Table.AddColumn(InsertCalendarMonth, "QuarterInCalendar", each "Q" & Number.ToText([QuarterOfYear]) & " " & Number.ToText([Year])),

InsertDayWeek = Table.AddColumn(InsertCalendarQtr, "DayInWeek", each Date.DayOfWeek([Date])),

InsertDayName = Table.AddColumn(InsertDayWeek, "DayOfWeekName", each Date.ToText([Date], "dddd"), **type** text),

InsertWeekEnding = Table.AddColumn(InsertDayName, "WeekEnding", each Date.EndOfWeek([Date]), **type** date),

InsertWeekNumber= Table.AddColumn(InsertWeekEnding, "Week Number", each Date.WeekOfYear([Date])),

InsertMonthnYear = Table.AddColumn(InsertWeekNumber,"MonthnYear", each [Year] \* 10000 + [MonthOfYear] \* 100),

InsertQuarternYear = Table.AddColumn(InsertMonthnYear,"QuarternYear", each [Year] \* 10000 + [QuarterOfYear] \* 100),

ChangedType1 = Table.TransformColumnTypes(InsertQuarternYear,{{"QuarternYear", Int64.Type},{"Week Number", Int64.Type},{"Year", **type** text},{"MonthnYear", Int64.Type}, {"DateInt", Int64.Type}, {"DayOfMonth", Int64.Type}, {"MonthOfYear", Int64.Type}, {"QuarterOfYear", Int64.Type}, {"MonthInCalendar", **type** text}, {"QuarterInCalendar", **type** text}, {"DayInWeek", Int64.Type}}),

InsertShortYear = Table.AddColumn(ChangedType1, "ShortYear", each Text.End(Text.From([Year]), 2), **type** text),

AddFY = Table.AddColumn(InsertShortYear, "FY", each "FY"&(**if** [MonthOfYear]>=FYStartMonth then Text.From(Number.From([ShortYear])+1) **else** [ShortYear]))

**in**

AddFY

**in**

fnDateTable

**step 5)** Click on **Done**

Now, an input form gets displayed which allows us to enter **StartDate**, **EndDate**and **FYStartMonth**(Financial Year Start Month).

Some companies may consider **FYStartMonth**as 1 while some consider it as 4.

Lets consider input  **Start Date**as 01-01-2014 and **End Date**as 31-12-2016 and

**FYStartMonth**as1.

Graphical user interface, text, application, email

Description automatically generated

**step 6)**Click on **Invoke**button.

Then date table gets created with the name "Invoked Function"

**step 7)**Lets rename the table as **DimDate**

After creating **DimDate**table, it is important to mark it as date table only then the power bi can recognise the newly created date table.

To do this:

Select **DimDate**table under **Fields**section and click on **Mark as date table**under**Table tools**section in power bi desktop.

Then a window appears as shown below:

Graphical user interface, text, application, email

Description automatically generated

Select column type as Date as shown below:

Graphical user interface, text, application, email

Description automatically generated

A message "Validated Successfully" gets displayed

Then Click on **ok.**

**Lets make Data Modelling Connections:**

Before proceeding with important and time intelligence measures, it is always necessary to check and make proper connections between the tables.

**Go to Model view:**

Here, we consider **Sales tables as fact table**and**other tables as dimension tables.**

**step 1)** **Make a connection from "Customer Data" table to "Sales" table:**

To do this:

Click and Drag **Customer Index**from **Customer Data**table and release onto **Customer Name** **Index**of **Sales**table i.e one-to-many connection from Customer Data table to Sales table.

**step 2) Make a connection from "Product Data" table to "Sales" table:**

To do this:

Click and Drag **Index**from **Product Data**table and release onto **Product Description** **Index**of **Sales**table i.e one-to-many connection from Product Data table to Sales table.

**step 3) Make a connection from "Regions" table to "Sales" table:**

To do this:

Click and Drag **Index**from **Regions**table and release onto **Delivery Region** **Index**of **Sales**table i.e one-to-many connection from Regions table to Sales table.

**step 4) Make a connection from "DimDate" table to "Sales" table:**

To do this:

Click and Drag **Date**from **DimDate**table and release onto **OrderDate**of **Sales**table i.e one-to-many connection from DimDate table to Sales table.

Lets keep **Metric Selection** table as a separate table as there is no use of this table.

**Lets create Time Intelligence Measures table:**

It is good practice to keep all the time intelligence measures at one place.

To do this:

Under "**Home**" section, click on "**Enter data**" and give it a name "**Time Intelligence Measures**" then click on "**Load**".

**Task 5) Need to have a details Performance LY for any selected Year.**

To know the last year performance for any selected year.

 we need to calculate **Sales Last Year** using below DAX formula:

Sales **Last** **Year** = CALCULATE([Total Sales],SAMEPERIODLASTYEAR(DimDate[Date]))

and then **Profit Last Year** using below DAX formula:

Profit **Last** **Year** = CALCULATE([Total Profit],SAMEPERIODLASTYEAR(DimDate[Date]))

Here, I used SAMEPERIODLASTYEAR function to calculate both sales and profit of last year.

We can use text table visual to display last year performance:

**To display last year performance for a selected year:**

We need to use **Year as slicer.**

**Last year Performance for 2015:**

Table

Description automatically generated

**Task 6) Comparing the Performance vs LY.**

Comparing performance vs Last year:

**Cumulative performance in terms of Sales:**

Chart, line chart

Description automatically generated

**Insights:**

 Current year sales shows better sales for products such as Brimmer, Nicel, Dimonies, Werrier, Linon, Qusec and loader compared to last year sales.

 Last year sales shows better sales for products  such as Assitus, Denycons, Penter, Unicoes, Ferrous, Channels and Carvetures compared to current year sales.

 Brimmer product is in high demand in terms of current and previous year sales.

 Carvetures product is in low demand in terms of current and previous year sales.

**Cumulative performance in terms of Profit:**

Chart, line chart

Description automatically generated

**Insights:**

 There is a huge improvement in terms of profit for products such as Nicel, Brimmer, Dimonies, Werrier and Linon compared to last year.

 The products such as Penter, Assitus, Unicoes, Denycons, Ferrous, Carvetures and Channels shows steady profit with little improvement.

 Nicel product brings highest profit  in terms of current and previous years.

 Channels product brings lowest profit  in terms of current and previous years.

**Task 7) Sales 2yrs ago for any Selected Year.**

We can calculate Sales 2yrs Ago using below DAX formula:

**Sales** 2yrs Ago = CALCULATE([Total Sales],DATEADD(DimDate[Date],-2,YEAR))

Here, I used DATEADD function to calculate 2 years ago sales.

In the DAX formula, -2 indicates 2 years back from a year.

Using **Year as slicer** to display sales 2 years ago for a selected year as shown below:

Table

Description automatically generated

**Task 8) It is important to know the moving average in terms of Profit & Sales.**

**Moving Average in terms of Sales:**

Sales Moving Average can be calculated using below DAX formula:

Sales Moving Average =

AVERAGEX(

**FILTER**(ALLSELECTED(DimDate),

DimDate[Date]<=MAX(DimDate[Date])),

[Total Sales])

Here, we apply the condition <= MAX of Date column of DimDate table using FILTER function with ALLSELECTED option on DimDate table and then we apply AVERAGEX function on Filter in terms of Total Sales.

**Moving Average in terms of Profit:**

Profit Moving Average can be calculated using below DAX formula:

Profit Moving Average =

AVERAGEX(

**FILTER**(ALLSELECTED(DimDate),

DimDate[Date]<=MAX(DimDate[Date])),

[Total Profit])

Here, we apply the condition <= MAX of Date column of DimDate table using FILTER function with ALLSELECTED option on DimDate table and then we apply AVERAGEX function on Filter in terms of Total Profit.

**We can use text table visual to display moving averages of sales and profit:**

Table

Description automatically generated

**Task 9) Quarter wise analysis is also important, may be slicer.**

we can use text table visual to show Quarter wise sales for each year using Quarter as a slicer.

Here, we drag and drop Year, Quarter in Date Hierarchy and Total Sales measure to data fields of

text table visual as shown below:

Table

Description automatically generated with low confidence

If we select Quarter 2 from the slicer. It displays quarter 2 sales of  all the three years as shown below:

Diagram

Description automatically generated with low confidence

**Task 10)  Total Sales, Profit and % Profit can be shown as combo with card and line chart.**

First, we need to import a custom visual that can show card with line chart .

**Here, lets import** a custom visual **Card with States by OKVIZ**using below steps:

**step 1)**Under **Visualizations**pane, click on**. . .**(dots) which says Get More Visuals when you hover your mouse on it.

**step 2)**Click on **Get more visuals**

Then a window pop ups. In the Search bar type "card with line" and click Enter.

**step 3)** Click on **Add**button of **Card with States by OKVIZ**visual

Graphical user interface, text, application, chat or text message

Description automatically generated

Then, the visual gets imported in to the power bi desktop under Visualizations pane.

**To show Total Sales, Profit and Profit% using this visual:**

Click on the custom card visual as shown below. Then drag and drop **Total Sales**and **Total Profit**

measures to **Measures(?)**field under Field section of Visualizations pane.

Here, I used **MonthInCalendar**as Trendline axis and **Profit Margin**as Tooltip.

**For Total Sales, trend line shows month wise sales with profit margin as tooltip:**

Text, whiteboard

Description automatically generated

**For Total Profit, trend line shows month wise profit with profit margin as tooltip:**

Graphical user interface

Description automatically generated with low confidence

**Task 11) Top Performer in terms of Product and Customer top 7 will give a clear idea.**

Lets first create a measure with the name Specific scenarios to keep all specific type of measures at one place for convenience.

To do this:

Under "**Home**" section, click on "**Enter data**" and give it a name "**Specific Scenarios**" then click on "**Load**".

Now, we need to right click on Specific Scenarios table and create the following measures:

**Calculating top 7 customers using below DAX formula:**

Top 7 Customers = CALCULATE([Total Sales],**FILTER**(**VALUES**('Customer Data'[Customer **Names**]),

**IF**(RANKX(**ALL**('Customer Data'[Customer **Names**]),[Total Sales],,**DESC**)<=7,[Total Sales],BLANK())))

Here, By applying RANKX  function on Customer Names in terms of Total Sales in Descending order. Since we want only top 7, the condition <=7 is applied and thus FILTER is applied using CALCULATE function on Total Sales.

**Top 7 Performers in terms of Customers:**

Table

Description automatically generated

Insight: Medline is the Top 1 Customer among all the customers

**Calculating top 7 products using below DAX formula:**

Top 7 Products = CALCULATE([Total Sales],**FILTER**(**VALUES**('Product Data'[Product **Name**]),

**IF**(RANKX(**ALL**('Product Data'[Product **Name**]),[Total Sales],,**DESC**)<=7,[Total Sales],BLANK())))

Here, By applying RANKX function on Product Name in terms of Total Sales in Descending order. Since we want only top 7, the condition <=7 is applied and thus FILTER is applied using CALCULATE function on Total Sales.

**Top 7 Performers in terms of Products:**

Table

Description automatically generated

Insight: Brimmer is the Top 1 Product among all the products

**Task 12) Performance in terms of region sales needs to workout and also needs to work on as what will be the best way to visualise it except Bar Graph.**

We can use **Map**visual under **Visualizations**pane.

To do this:

Go to View Section, Select **Map**visual (globe icon) under **Visualizations**pane.

**The steps I followed under Visualization pane to show region sales:**

 Territory column is brought to Location field

 Total Sales measure is brought to Size field

 Total Profit and Profit Margin measures are brought to Tooltips field

Now, the map shows region sales for each Territory of Australia as shown below:

Map

Description automatically generated

**Drill-through** is enabled and applied to **Territory**category to show city wise sales for each Territory.

**For example:**

When right-clicked on **Queensland**Territory to drill through. City wise sales page shows City wise sales of Queensland as shown below:

Chart

Description automatically generated

**Data Model View:**

Graphical user interface, diagram

Description automatically generated

**Dashboard:**

Diagram

Description automatically generated with low confidence

**You can view my short demo on this project reports and dashboard using below link:**