

Data Analytics

Advance Analytics (Road map)

Step 1- Change - Over - time (trends)

2- Cumulative Analysis

3- Performance Analysis

4- Part - to - Whole (Proportional)

5- Data Segmentation

6- Reporting

~~Data Analytics~~

~~Data~~ Advanced Data Analytics.

1. Changes Over time Analysis.

[Measure] By [Date Dimension]

Total Sales by year

Average Cost by month

Ex:- select date_trunc (Year, order_date) as order_date,
Sum (Sales amount) total_sales, Count (Distinct customer_id)
total_customers, Sum (quantity) total_quantity
From gold.fact-sales

Group by date_trunc (Year, order_date)

Order by date_trunc (Year, order_date)

2. Cumulative Analysis.

[Cumulative Measure] By [Date Dimension]

Running total Sales by year

Moving Average of Sales by month

Ex:- Select order_date, total_sales,

Sum (total_sales) over (Order by order_date) as Running_T,

Avg (Avg-price) over (Order by order_date) as moving_Avg

From (Select Sum

Sum date_trunc (month, ^{order_date} date) as order_date,

Sum (Sales amount) as total_sales,

Avg (price) as Avg-price

From gold.fact-sales

where order-date is not null
group by date-to-enc (month, order-date)
);

3. Performance Analysis

current[Messure] :- Target[Messure]

like :- Current Sales - Average Sales

Current Year Sales - Previous Year Sales

Current Sales - Lowest Sales

- Analyze the yearly performance of products by comparing their sales to both the average sales performance of the product and the previous year's sales with yearly-product sales as

Select Year(orders.order-date) as order-date,
p.product-name, Sum(s.sale-amount) as
current-sales.

From gold.fact-sales as s

left join gold.dim-products as p

On p.product-key = s.product-key

where s.order-date is not null

Group by p.product-name, Year(s.order-date)
)

Select order-date, product-name, current-sales,
~~current-sales~~ - Avg (current-sales) over (partition by productname)
as Avg-sales;

X { Case when Avg (current-sales) over (partition by productname)
> 0 then 'Above avg'
When Avg (current-sales) over (partition by productname)
< 0 then 'below avg'
Else 'Avg' end
end * Avg-change;

currentsales - Avg (current-sales) over (partition by
product-name). as diff-avg,
Case when currentsales - Avg (current-sales) over (partition
by productname) then 'Above avg'
when current-sales - Avg (current-sales) over (partition by
product-name) then 'below avg'
else 'Avg'
end * Avg-change;

lag (current-sales) over (order by partition by pname
order by order-date) as py-sales,
currentsales - lag (current-sales) over (partition by
p-name order by order-date) as pdiff-py.
Case when " > 0 then 'Increase'
when " < 0 then 'Decrease'
else 'onchange'
end py-sales-change;

From yearly-product-sales

Order by product-name, orderdate.

4. Part - to - Whole

$(\text{Measure} / \text{Total Measure}) * 100$ By [Dimension]

like $(\text{Sales} / \text{Total Sales}) * 100$ By category.

$(\text{Quantity} / \text{Total Quantity}) * 100$ by country.

which categories contribute the most to overall sales.

With category-sales

Select p.category, sum(s.sales-amount) as totalsales

From gold.fact-sales s

left join gold.dim-products

on p.product-key = s.product-key

Group by p.category

Select category, ~~sum~~ totalsales,
sum(totalsales) over() overallsales;

concat(Round(Cast(totalsales as float) / sum(totalsales)) * 100,
2), '%') as percent_percentage_sales

From category-sales

Order by ~~category~~ totalsales desc

5. Data Segmentation

[Measure] by [Measure]

like - Total product by Sales Range
Total Customers by Age

- Segment products into cost ranges and count how many products falls into each segment.
with product segment as (

Select product-key, product-name,

Case when cost < 100 then 'Below'

when cost between 100 and 500 then '100-500'

when cost between 500 and 1000 then '500-1000'

when cost > 1000 then 'Above'

else 'Above 1000'

end cost-range

From gold.dim-products)

Select cost-range, count(product-key) as total-pro

From product-segment

Group by total-pro cost-range