**PROMPT - GIVE ME BRIEF DOCUMENTATION ON NORMALIZATION IN POWER BI**

**Normalization in Power BI**

**1. What is Normalization?**

In data analysis, **normalization** means adjusting or transforming data to a common scale without distorting differences in the values.  
In **Power BI**, normalization is often used to:

* Compare data with different ranges (e.g., sales vs. profit margins).
* Standardize data for machine learning or statistical modeling.
* Make visualizations more meaningful by removing scale bias.

**2. Types of Normalization in Power BI**

There is no single "Normalize" button in Power BI, but you can achieve normalization using **Power Query** or **DAX calculations**.

**(a) Min-Max Normalization**

Transforms data to a fixed range, usually **0 to 1**.  
Formula:

Normalized Value = (Value - MIN(Column)) / (MAX(Column) - MIN(Column))

**Use case:** To scale features like revenue, profit, and cost into a comparable 0–1 range.

**(b) Z-Score Normalization (Standardization)**

Centers the data around **mean = 0** with a **standard deviation = 1**.  
Formula:

Z-Score = (Value - AVERAGE(Column)) / STDEV(Column)

**Use case:** Useful for identifying outliers and statistical analysis.

**(c) Log Transformation**

Applies logarithmic scaling to compress large ranges.  
Formula (in DAX):

Log Value = LOG(Column, 10)

**Use case:** To handle skewed data or extreme variations (e.g., population data).

**(d) Row-Level Normalization**

If you want to normalize values **within each row** (e.g., sales contribution per product in a row), you can calculate:

Row Normalized Value = Column Value / SUM(Row Values)

**3. How to Apply Normalization in Power BI**

**🔹 Using Power Query**

* Go to **Transform Data** → **Power Query Editor**.
* Use transformations like **Standardize, Min-Max scaling, or custom formulas**.
* Example: Add a custom column → apply formula for Min-Max or Z-Score.

**🔹 Using DAX**

* Create a **new measure** or **calculated column** with normalization formulas.
* Example for Min-Max:

Normalized Sales =

VAR MinSales = CALCULATE(MIN(Sales[Amount]), ALL(Sales))

VAR MaxSales = CALCULATE(MAX(Sales[Amount]), ALL(Sales))

RETURN

DIVIDE(Sales[Amount] - MinSales, MaxSales - MinSales)

**4. Why Normalization is Important in Power BI**

* Makes **comparisons fair** across different scales.
* Helps in **data modeling** for AI/ML visuals.
* Improves **visual readability** (charts, KPIs).
* Reduces **outlier impact** in reports.

let’s make normalization in Power BI feel concrete. Below are **five practical examples** with formulas, visuals you’d use, and step-by-step notes (DAX + Power Query). I’ll assume a simple table **Sales** with columns: Date, Region, Product, OrderID, Quantity, Amount.

**1) Min–Max Normalization (0–1) for fair comparison**

**Goal:** Put Amount on a 0–1 scale so you can compare products/regions fairly.

**DAX (responds to slicers/filters)**

Normalized Amount :=

VAR MinAmt =

CALCULATE( MIN ( Sales[Amount] ), ALL ( Sales ) )

VAR MaxAmt =

CALCULATE( MAX ( Sales[Amount] ), ALL ( Sales ) )

RETURN

DIVIDE ( Sales[Amount] - MinAmt, MaxAmt - MinAmt )

**How it works**

* ALL(Sales) removes filters so the **global** min/max are used (good for ranking everything against the full dataset).
* DIVIDE safely avoids divide-by-zero (returns BLANK if Max=Min).

**Variation – normalize within current filters (e.g., within the selected year):**

Normalized Amount (Ctx) :=

VAR MinAmt = MIN ( Sales[Amount] )

VAR MaxAmt = MAX ( Sales[Amount] )

RETURN DIVIDE ( Sales[Amount] - MinAmt, MaxAmt - MinAmt )

This version uses the **current filter context** (whatever slicers are active).

**Where to use it**

* Put Product on a **bar chart** with Normalized Amount as value to compare relative performance without scale bias.

**2) Z-Score (Standardization) to spot outliers**

**Goal:** Center values around 0 with standard deviation 1; great for outlier detection.

**DAX – Z-score across the whole model**

ZScore Amount :=

VAR MeanAmt =

CALCULATE ( AVERAGE ( Sales[Amount] ), ALL ( Sales ) )

VAR StdAmt =

CALCULATE ( STDEV.P ( Sales[Amount] ), ALL ( Sales ) )

RETURN DIVIDE ( Sales[Amount] - MeanAmt, StdAmt )

**Per-group Z-score (e.g., within each Region)**

ZScore Amount by Region :=

VAR MeanAmt =

CALCULATE ( AVERAGE ( Sales[Amount] ), ALLEXCEPT ( Sales, Sales[Region] ) )

VAR StdAmt =

CALCULATE ( STDEV.P ( Sales[Amount] ), ALLEXCEPT ( Sales, Sales[Region] ) )

RETURN DIVIDE ( Sales[Amount] - MeanAmt, StdAmt )

**Where to use it**

* **Scatter chart** with Product points: X = ZScore Amount by Region, Y = another standardized metric (e.g., margin). Points beyond ±2 are potential outliers.

**3) Row-level Normalization (Percent of each Order)**

**Goal:** Within an order, show each product’s share of the order total (adds to 100% per row group).

**DAX (calculated column or measure)**

Order Share :=

DIVIDE (

Sales[Amount],

CALCULATE ( SUM ( Sales[Amount] ), ALLEXCEPT ( Sales, Sales[OrderID] ) )

)

**Explanation**

* ALLEXCEPT keeps only OrderID as the grouping key: denominator = total for that order.
* Result is **0–1**; format as **Percentage**.

**Where to use it**

* **Matrix**: Rows = OrderID, Columns = Product, Values = Order Share. You’ll see product mix per order (each row sums to ~100%).

**4) Log Transform for skewed data**

**Goal:** Compress huge ranges (e.g., some invoices are 10× larger than others) so patterns show up.

**DAX (safe for zeros)**

Log Amount := LN ( Sales[Amount] + 1 )

* Use LN (natural log). Adding **1** avoids LN(0) errors.

**Where to use it**

* **Histogram** (via a custom visual) or **line chart** over time using Log Amount to reveal trends hidden by extreme values.

**5) Power Query: Group-wise Min–Max Normalization (no DAX)**

**Goal:** Normalize Amount **inside Power Query** per Region, then load a clean, normalized column to the model.

**Steps (UI)**

1. **Home ▸ Transform Data** to open Power Query.
2. **Group By** Region to compute MinAmount and MaxAmount.
3. **Merge** the grouped table back to the original on Region.
4. Add a **Custom Column**:  
   ( [Amount] - [MinAmount] ) / ( [MaxAmount] - [MinAmount] )
5. Replace errors (if Max=Min, result is null), optional.
6. Remove helper columns; **Close & Apply**.

**M code sketch (for reference)**

let

Source = Excel.Workbook(File.Contents("..."), null, true){[Name="Sales"]}[Content],

ChangedTypes = Table.TransformColumnTypes(Source,{{"Region", type text}, {"Amount", type number}}),

StatsByRegion = Table.Group(ChangedTypes, {"Region"},

{{"MinAmount", each List.Min([Amount]), type number},

{"MaxAmount", each List.Max([Amount]), type number}}),

Merged = Table.NestedJoin(ChangedTypes, {"Region"}, StatsByRegion, {"Region"}, "RG", JoinKind.LeftOuter),

Expanded = Table.ExpandTableColumn(Merged, "RG", {"MinAmount", "MaxAmount"}),

AddedNorm = Table.AddColumn(Expanded, "NormAmount", each

try ([Amount] - [MinAmount]) / ([MaxAmount] - [MinAmount]) otherwise null, type number)

in

AddedNorm

**Where to use it**

* Any visual directly from the model—no DAX required. Great when you want consistent, pre-computed normalization.

**6) Bonus: “Index to 100” (Rebasing a time series)**

**Goal:** Compare different products/regions over time by indexing each to **100 at the first period** in the current filter context.

**DAX**

Index to 100 :=

VAR FirstAmt =

CALCULATE (

SUM ( Sales[Amount] ),

DATEADD ( 'Date'[Date], 0, DAY ),

KEEPFILTERS ( 'Date'[Date] = CALCULATE ( MIN ( 'Date'[Date] ), ALLSELECTED ( 'Date'[Date] ) ) )

)

VAR CurrAmt = SUM ( Sales[Amount] )

RETURN 100 \* DIVIDE ( CurrAmt, FirstAmt )

**Simpler version if you have a marked Date table**

Index to 100 (Simple) :=

VAR FirstAmt =

CALCULATE ( SUM ( Sales[Amount] ), FILTER ( ALLSELECTED ( 'Date'[Date] ), 'Date'[Date] = MIN ( 'Date'[Date] ) ) )

RETURN 100 \* DIVIDE ( SUM ( Sales[Amount] ), FirstAmt )

**Where to use it**

* **Line chart**: Axis = Date, Legend = Product, Values = Index to 100. You’ll see relative growth paths from the same starting base.

**When to choose a Measure vs a Calculated Column**

* **Measure**: reacts to slicers, time intelligence, and visuals. Use for normalization that must respect filters (most cases).
* **Calculated column**: fixed at refresh time; use when you want a static normalized value or need row context.

**Practical Pitfalls & Tips**

* **Divide-by-zero**: always use DIVIDE(numerator, denominator) or wrap with IF(denominator=0, BLANK(), …).
* **Context control**:
  + Use ALL() to normalize against the entire dataset.
  + Use ALLEXCEPT(Table, GroupColumn) to normalize per group.
  + Use REMOVEFILTERS() when you want to clear filters more explicitly.
* **Outliers**: For Min–Max, extreme outliers will squeeze most values near 0. Consider **winsorizing** (capping) before normalization.
* **Formatting**: Z-scores should be **Decimal Number**; Min–Max and shares as **Percentage** if you want 0–100%.
* **Performance**: Measures recompute; if your dataset is huge and the normalization is stable, consider doing it in **Power Query**.

✅ **In summary:**  
Normalization in Power BI means transforming data into a comparable scale using methods like **Min-Max scaling, Z-score, Log transformation, or row-based normalization**. You can implement it through **Power Query** or **DAX formulas** depending on your use case.