1. IF (RANKX(ALL(Online_Courses [Category]),CALCULATE(AVERAGE(Online_Courses[Number of viewers])))<6,CALCULATE(AVERAGE(Online_Courses[Number of viewers])),BLANK())

- RANKX is a DAX function used to rank items (like rows in a table) based on a specific value or calculation.
- It compares values and assigns a rank (1st, 2nd, 3rd, etc.) to each item.
- The smaller the value, the higher the rank (or vice versa if sorting in descending order).

RANKX Syntax:

```DAX
RANKX(
    table,              -- The table or set of values to rank.
    expression,         -- The value or calculation used for ranking.
    [value],            -- Optional: A specific value to compare.
    [order],            -- Optional: Sorting order (ASC or DESC).
    [ties]              -- Optional: How to handle tied ranks.
)
```

- table: The dataset you are ranking. For example, it could be a list of instructors.
- expression: The value you use to rank, like average rating or total sales.
- [value]: You can specify a single value to check its rank (optional).
- [order]: You can choose ASC (ascending) or DESC (descending) for ranking (optional).
- [ties]: Specifies whether to skip or assign the same rank for ties (optional).

Full Function Breakdown:

```
Instructor Rating =
IF(
    RANKX(
        ALL(Instructors[Instructors]),
        CALCULATE(AVERAGE(Online_Courses[Rating]))
    ) <= 3,
    CALCULATE(AVERAGE(Online_Courses[Rating])),
    BLANK()
)
```

RANKX Part:

- **ALL(Instructors[Instructors]):** This ensures we rank all instructors, ignoring any filters on the column.
- **CALCULATE(AVERAGE(Online_Courses[Rating])):** Calculates the average rating for each instructor. This value is used for ranking.
- The RANKX function assigns a rank to each instructor based on their average rating.

2. IF Condition:

- The IF checks if the instructor's rank is less than or equal to 3 (<= 3).
- If the rank is in the **top 3**, it calculates and shows the average rating of the instructor.

3. BLANK() Default:

- If the rank is not in the top 3, it returns a blank value instead of a number.

Simple Explanation of the Function:

- The function ranks all instructors by their average course rating.
- If an instructor is in the top 3, their average rating is displayed.
- If they are not in the top 3, nothing (blank) is shown.

==========

```
if Text.Contains(Text.Lower([Duration]), "month") then
    Number.FromText(Text.Select(Text.Lower([Duration]), {"0".."9"})) * 60
else if Text.Contains(Text.Lower([Duration]), "hour") then
    Number.FromText(Text.Select(Text.Lower([Duration]), {"0".."9"}))
else if Text.Contains(Text.Lower([Duration]), "minute") then
    Number.FromText(Text.Select(Text.Lower([Duration]), {"0".."9"})) / 60
else
    200
```

1. **Text.Contains**

- This function checks whether a specific word or text exists in a given column or string.

Syntax : Text.Contains(text, substring)

- text: The full text you are checking.
- substring: The specific word you want to find.
- Returns true if the substring is found, otherwise false.

Example:

- Text.Contains("5 hours to complete", "hour") → true
- Text.Contains("15 minutes to complete", "hour") → false

2. **Text.Lower**

- Converts all text to lowercase to make comparisons case-insensitive.

Syntax: Text.Lower(text)

- text: The string you want to convert to lowercase.

Example:

- Text.Lower("MONTHS") → "months"
- Text.Lower("Hours") → "hours"

3. **Number.FromText**

Converts numeric text into a number.

Syntax: Number.FromText(text)

text: The string containing numeric values.

Example:

- Number.FromText("123") → 123
- Number.FromText("45.67") → 45.67

4. **Text.Select**

Extracts specific characters from a text based on a list of allowed characters.

Syntax: Text.Select(text, {"allowed_characters"})

- text: The string you are processing.
- allowed_characters: A list of characters you want to keep.

Example:

- Text.Select("15 hours to complete", {"0".."9"}) → "15"
- Text.Select("3 months", {"0".."9"}) → "3"

What Does the Full Function Do?

Step-by-Step Explanation:

1. **Check for "month":**

- The function first checks if the text in [Duration] contains the word "month" (case-insensitive).
- If "month" is found:
- Extract the numeric part using Text.Select.
- Multiply the number by 60 (to convert months to hours).

Example:

- Input: "3 months to complete"
- Extract 3 → 3 * 60 = 180.

2. **Check for "hour":**

- If "month" is not found, it checks for the word "hour."
- If "hour" is found:
- Extract the numeric part using Text.Select.
- Return the number as it is (since it's already in hours).

Example:

- Input: "2 hours to complete"
- Extract 2 → Return 2.

3. **Check for "minute":**

- If neither "month" nor "hour" is found, it checks for the word "minute."
- If "minute" is found:
- Extract the numeric part using Text.Select.
- Divide the number by 60 (to convert minutes to hours).

Example:

- Input: "90 minutes to complete"
- Extract 90 → 90 / 60 = 1.5.

4. **Default Case:**

- If none of the keywords ("month," "hour," or "minute") are found, it returns 200 as a default value.

Example:

- Input: "Unknown duration"
- Return 200.

Summary:

- This function processes a Duration column containing phrases like "3 months to complete" or "2 hours to complete."
- It converts durations into hours:
- Months → Multiply by 60.

- Minutes → Divide by 60.
- Hours → Use as-is.
- If no valid duration is found, it defaults to 200.

==========

The **Text.Split** function in M code is used to split a text string into smaller parts (called "substrings") based on a specific character or delimiter.

Text.Split(text, delimiter)

- text: The string you want to split.
- delimiter: The character(s) used to divide the text.

How It Works

- It looks at the text and cuts it into pieces wherever the delimiter appears.

Examples

1. **Splitting a Sentence**

- Text.Split("Hello World", " ")
- Input: "Hello World"
- Delimiter: " " (space)
- Output: {"Hello", "World"} (a list of two parts)

2. Splitting Based on a Comma

- Text.Split("apple,banana,orange", ",")
- Input: "apple,banana,orange"
- Delimiter: ","
- Output: {"apple", "banana", "orange"}

3. **Splitting by a Specific Word**

- Text.Split("2 hours to complete", "hours")
- Input: "2 hours to complete"
- Delimiter: "hours"
- Output: {"2 ", " to complete"}

What Happens After Splitting?

- The result is always a list of parts.
- You can use the list to get specific pieces of the text or manipulate them further.
-

Text.Length

What It Does:

- Calculates the total number of characters in a given text string.

Syntax: Text.Length(text as nullable text) as nullable number

- text: The text input whose length you want to find.
- Returns: The number of characters in the input string.

Example:

- Text.Length("Python, SQL, Power BI")
- Output: 21 (counts every letter, comma, and space in the string).

Text.Replace

What It Does:

- Replaces all occurrences of a specific substring in a text with another string

Syntax: Text.Replace(text as nullable text, oldValue as text, newValue as text) as nullable text

- text: The input text where you want to perform replacements.
- oldValue: The substring you want to replace.
- newValue: The string you want to replace it with.
- Returns: A new string with the replacements made.

Example:

- Text.Replace("Python, SQL, Power BI", ",", "")
- Output: "Python SQL Power BI" (removes commas).

The Full Code:

```
Text.Length([Skills]) - Text.Length(Text.Replace([Skills], ",", ""))
```

Text.Length([Skills]):

- Finds the total length of the text in the [Skills] column, including commas and spaces.

Text.Replace([Skills], ",", ""):

- Removes all commas from the text in the [Skills] column.

Text.Length(Text.Replace([Skills], ",", "")):

- Finds the length of the text in the [Skills] column after removing commas.

Subtraction:

- Subtracts the length of the string without commas from the original length.

- This difference gives the number of commas in the text.

Example:

If the [Skills] column contains:

"Python, SQL, Power BI"

Step 1: Text.Length([Skills]) = 21

(Counts all characters including commas and spaces).

Step 2: Text.Replace([Skills], ",", "") = "Python SQL Power BI"

(Removes commas).

Step 3: Text.Length(Text.Replace([Skills], ",", "")) = 19

(Counts all characters except commas).

Step 4: 21 - 19 = 2

The result is 2, indicating there are 2 commas in the string.

Purpose:

This code is often used to count specific characters (like commas) in a string by comparing the original length with the length after removing those characters.