Lesson 12

Topic: Text, Informational and Logical Functions
Prerequisites: Connect SQL Server to Power BI desktop and import DimCustomer table from AdventureWorksDW database (2019)
Questions:

Basic Level (1–10)

- **Concatenate First and Last Name**

  You need to create a **calculated column** that joins the first and last names from the DimCustomer table.

  DAX Formula:

  You can create a new column using the following DAX expression:

  dax

  Full Name = DimCustomer[FirstName] & " " & DimCustomer[LastName]

  OR using the CONCATENATE function (less preferred for longer chains of text):

  dax

  Full Name = CONCATENATE(DimCustomer[FirstName], " " & DimCustomer[LastName])

Note: The & operator is more flexible and readable for combining multiple strings compared to CONCATENATE.

Explanation:

- DimCustomer[FirstName] and DimCustomer[LastName] are column references.
- " " adds a space between the first and last names.
- & joins the strings together.

Where to add this:

In Power BI Desktop:

1. Go to the **Data View**.
2. Select the **DimCustomer** table.
3. Click on **New Column**.
4. Paste the DAX code above.

- **Convert Email Address to Uppercase**

  We want to create a calculated column that converts the EmailAddress field in the DimCustomer table to all uppercase letters.

  DAX Formula:

  dax

  Email Upper = UPPER(DimCustomer[EmailAddress])

  Explanation:

- UPPER() is a Text function in DAX.
- It takes a text string (in this case, the email address) and returns it in uppercase.

  Steps to Create:

1. In Power BI Desktop, go to the Data View.
2. Select the DimCustomer table.
3. Click New Column on the ribbon.
4. Paste the formula above.

- **Extract First 3 Characters from First Name**

  You'll create a calculated column that returns only the first 3 letters of each customer's FirstName from the DimCustomer table.

  DAX Formula:

dax

First 3 Letters = LEFT(DimCustomer[FirstName], 3)

Explanation:

- LEFT(text, num_chars) is a text function that returns the leftmost num_chars characters from a string.
- In this case:
    - text is DimCustomer[FirstName]
    - num_chars is 3

How to Implement:

1. Go to the Data View in Power BI Desktop.
2. Click on the DimCustomer table.
3. Select New Column.
4. Paste the DAX code above.

- **Count Characters in Last Name**

  You need to create a calculated column that counts the number of characters in the LastName column of the DimCustomer table.

  DAX Formula:

  dax

  Last Name Length = LEN(DimCustomer[LastName])

  Explanation:

- LEN() is a Text function in DAX.
- It returns the number of characters in a text string (including spaces if there are any).

  Steps:

1. In Power BI Desktop, go to the Data View.
2. Click on the DimCustomer table.

3. Click New Column.
4. Paste the formula above.

- **Convert First Name to Lowercase**

  You need to create a calculated column that converts the FirstName in the DimCustomer table to all lowercase letters.

  DAX Formula:

  dax

  First Name Lower = LOWER(DimCustomer[FirstName])

  Explanation:

- LOWER() is a Text function in DAX.
- It takes a text string and converts all letters to lowercase.

Implementation:

1. Go to the Data View in Power BI Desktop.
2. Select the DimCustomer table.
3. Click on New Column.
4. Paste the formula above.

- **Trim Spaces in EnglishEducation**

  You need to create a calculated column that removes any leading or trailing spaces from the EnglishEducation column in the DimCustomer table.

  DAX Formula:

  dax

  Education Trimmed = TRIM(DimCustomer[EnglishEducation])

  Explanation:

- TRIM() is a DAX text-cleaning function.
- It removes:
  - Leading spaces (at the beginning)

- Trailing spaces (at the end)
- Extra internal spaces are not removed (only one space is kept between words).

How to Use:

1. Go to Data View in Power BI Desktop.
2. Select the DimCustomer table.
3. Click New Column.
4. Enter the formula above.

- **Repeat '*' Character Equal to Length of First Name**

  You need to create a calculated column that returns a string of * characters where the number of * is equal to the length of each customer's First Name.

  DAX Formula:

  dax

  Stars = REPT("*", LEN(DimCustomer[FirstName]))

  Explanation:

- LEN(DimCustomer[FirstName]): Calculates the number of characters in the first name.
- REPT("*", number): Repeats the * character as many times as specified by the second argument.

  Example:

  If the first name is John (4 characters), the column will return: ****

  Steps:

1. Go to Data View in Power BI Desktop.
2. Select the DimCustomer table.
3. Click on New Column.
4. Paste the DAX formula above.

- **Get Last 4 Characters of Phone Number**

  You'll create a calculated column that returns the last 4 digits of the Phone number from the DimCustomer table.

  DAX Formula:

  dax

  Last 4 Digits = RIGHT(DimCustomer[Phone], 4)

  Explanation:

- RIGHT(text, num_chars) is a Text function in DAX.
- It extracts the rightmost num_chars characters from a given text string.

  Example:

  If the phone number is 123-456-7890, this column will return 7890.

Steps:

1. In Power BI Desktop, go to Data View.
2. Select the DimCustomer table.
3. Click on New Column.
4. Paste the formula above.

- **Format YearlyIncome to Currency with 2 Decimals**

  You need to create a calculated column that formats the YearlyIncome column from the DimCustomer table as currency with 2 decimal places.

  DAX Formula:

  dax

  Income Formatted = FORMAT(DimCustomer[YearlyIncome], "₹#,##0.00")

Replace ₹ with $, €, or your preferred currency symbol.

Explanation:

- FORMAT(number, format_string) returns a number as a formatted text string.
- "#,##0.00":

- , adds thousands separator
- 0.00 ensures two decimal places

Example:

If YearlyIncome is 75000, it will display as:

- ₹75,000.00 or $75,000.00 depending on the symbol you use.

Steps:

1. In Data View, click on the DimCustomer table.
2. Click New Column.
3. Paste the formula above and choose the appropriate currency.

- **Check If FirstName and LastName Are Exactly the Same**

  You need to create a calculated column that checks whether the FirstName and LastName columns are exactly equal, including case sensitivity.

  DAX Formula:

  dax

  Same Name? = IF(DimCustomer[FirstName] = DimCustomer[LastName], "Yes", "No")

  Explanation:

- The = operator in DAX compares text exactly, including case.
- IF(condition, result_if_true, result_if_false) returns "Yes" if the first and last names are the same, otherwise "No".

  Optional (Case-Insensitive Check):

  If you want a case-insensitive comparison, use:

  dax

  CopyEdit

  Same Name (Case-Insensitive)? = IF(UPPER(DimCustomer[FirstName]) = UPPER(DimCustomer[LastName]), "Yes", "No")

  Steps:

1. Go to Data View in Power BI Desktop.
2. Select the DimCustomer table.
3. Click New Column.
4. Paste one of the formulas above.

Intermediate Level (11–20)

- **Find If 'Manager' Appears in Occupation (Case Sensitive)**

  You need to check whether the exact word 'Manager' (case-sensitive) appears anywhere in the Occupation column of the DimCustomer table.

  DAX Formula:

  dax

  Is Manager? = IF(SEARCH("Manager", DimCustomer[Occupation], 1, 0) > 0, "Yes", "No")

However, note: SEARCH() is not case-sensitive.

For Case-Sensitive search, use FIND() instead:

dax

Is Manager? = IF(FIND("Manager", DimCustomer[Occupation], 1, 0) > 0, "Yes", "No")

Explanation:

- FIND(subtext, fulltext, start_position, not_found_value):
  - Returns position of "Manager" if found (starting from position 1).
  - Returns 0 if not found (because we set not_found_value to 0).
- IF(...) > 0 means "was it found?"

Example:

- If Occupation = "Sales Manager" → Result: "Yes"
- If Occupation = "sales manager" → Result: "No" (because of case-sensitivity)

- **Search for 'graduate' in EnglishEducation (Case Insensitive)**

  You want to check if the word "graduate" appears anywhere in the EnglishEducation column, ignoring case sensitivity.

  DAX Formula:

  dax

  Has Graduate? = IF(SEARCH("graduate", DimCustomer[EnglishEducation], 1, 0) > 0, "Yes", "No")

  Explanation:

- SEARCH() performs a case-insensitive search.
- "graduate" will match:
  - "Graduate Degree"
  - "Partial Graduate"
  - "Some Graduate School"
- If the substring is found, it returns the position (1 or higher).
- If not found, it returns 0 because of our custom not_found_value.

  Steps:

1. In Power BI, go to Data View.
2. Select the DimCustomer table.
3. Click on New Column.
4. Paste the formula above.

- **Extract Characters 3–7 from First Name**

  You need to extract characters from position 3 to 7 in the FirstName column. This means 5 characters total, starting from the 3rd character.

  DAX Formula:

  dax

  First Name 3–7 = MID(DimCustomer[FirstName], 3, 5)

Explanation:

- MID(text, start_position, num_chars) extracts a substring:
  - From FirstName
  - Starting at position 3
  - For a total of 5 characters (i.e., positions 3, 4, 5, 6, and 7)
- If the first name is shorter than 7 characters, it returns what's available.

Example:

- FirstName = "Michael" → Result: "chae"
- FirstName = "Eva" → Result: "a" (only the available part)

- **Replace Area Code in Phone Number with 'XXX'**

  Assuming the area code is the first 3 characters of the phone number, you want to replace those with 'XXX'.

  DAX Formula:

  dax

  Phone Masked = "XXX" & MID(DimCustomer[Phone], 4, LEN(DimCustomer[Phone]) - 3)

  Explanation:

- "XXX": Hardcoded replacement for the area code.
- MID(DimCustomer[Phone], 4, LEN(...) - 3):
  - Extracts everything from 4th character to the end.
  - LEN(...) - 3 ensures the remaining length is dynamic.
- The result is the original phone number with the first 3 characters replaced.

Example:

- Phone = "123-456-7890" → Result: "XXX-456-7890"

- **Format BirthDate as 'DD-MM-YYYY'**

You need to display the BirthDate from the DimCustomer table in the DD-MM-YYYY format as a text value.

DAX Formula:

dax

Formatted BirthDate = FORMAT(DimCustomer[BirthDate], "DD-MM-YYYY")

Explanation:

- FORMAT() converts a date or number into a formatted text string.
- "DD-MM-YYYY" returns day, month, and full year in two-digit/two-digit/four-digit format.

Example:

- BirthDate = 1980-02-15 → Result: "15-02-1980"

- **Create Initial + Last Name Format (e.g. J.Smith)**

  You want to generate a name format using the first letter of the first name, followed by a dot, and then the last name — like "J.Smith".

  DAX Formula:

  dax

  Initial + Last Name = LEFT(DimCustomer[FirstName], 1) & "." & DimCustomer[LastName]

  Explanation:

- LEFT(DimCustomer[FirstName], 1): Gets the first character of the first name.
- & "." &: Adds the dot between initial and last name.
- DimCustomer[LastName]: Appends the full last name.

  Example:

- FirstName = "John", LastName = "Smith" → Result: "J.Smith"

- **Capitalize First Letter of FirstName, Lowercase the Rest**

  You need to transform the FirstName so that:

- The first letter is uppercase
- The rest of the name is lowercase

  DAX Formula:

  dax

  Proper FirstName = UPPER(LEFT(DimCustomer[FirstName], 1)) &
  LOWER(MID(DimCustomer[FirstName], 2,
  LEN(DimCustomer[FirstName]) - 1))

  Explanation:

- UPPER(LEFT(..., 1)): Capitalizes the first character.
- LOWER(MID(..., 2, ...)): Converts the rest of the string to lowercase.
- This formula ensures consistent formatting even if the original name is in all caps or mixed case.

Example:

- "jOHN" → "John"
- "ALICE" → "Alice"
- "bob" → "Bob"

- **Substitute Dashes with Spaces in Phone**

  You want to replace all dash (-) characters in the Phone column with spaces.

  DAX Formula:

  dax

  Phone with Spaces = SUBSTITUTE(DimCustomer[Phone], "-", " ")

  Explanation:

- SUBSTITUTE(text, old_text, new_text) replaces all instances of old_text with new_text.
- Here, every "-" in the phone number will be replaced with a " " (space).

Example:

- "123-456-7890" → "123 456 7890"

- **Convert BirthDate Year to Numeric Using VALUE**

  You need to extract the year part from the BirthDate column and convert it into a numeric value using the VALUE() function.

  DAX Formula:

  dax

  Birth Year (Numeric) = VALUE(YEAR(DimCustomer[BirthDate]))

  Explanation:

- YEAR(DimCustomer[BirthDate]) extracts the year from the date (e.g., 1985).
- VALUE(...) ensures the result is a numeric data type (though YEAR already returns a number, VALUE enforces explicit conversion in case you're working with text-based results or need numeric typing).

  Example:

- BirthDate = 1990-08-15 → Birth Year (Numeric) = 1990

- **Show YearlyIncome Rounded to 1 Decimal Without Commas**

  You need to display YearlyIncome:

- Rounded to 1 decimal place
- Without thousands separators (commas)

  DAX Formula:

  dax

  Income Rounded = FORMAT(ROUND(DimCustomer[YearlyIncome], 1), "0.0")

  Explanation:

- ROUND(DimCustomer[YearlyIncome], 1): Rounds income to 1 decimal place.

- FORMAT(..., "0.0"): Ensures no commas and shows exactly one digit after the decimal point.

  Example:

- 72500.567 → 72500.6
- 104000.0 → 104000.0 (no comma)

Advanced Level (21–30)

- **Customer Code: First 2 Letters of LastName + Last 2 of CustomerKey**

  Format:

First 2 letters of LastName + Last 2 digits of CustomerKey

  DAX Formula:

  dax

  Customer Code =

  LEFT(DimCustomer[LastName], 2) &

  RIGHT(FORMAT(DimCustomer[CustomerKey], "00"), 2)

  Explanation:

- LEFT(DimCustomer[LastName], 2): Gets the first 2 letters of the last name.
- FORMAT(DimCustomer[CustomerKey], "00"): Ensures CustomerKey has at least 2 digits (pads with leading zeroes if needed).
- RIGHT(..., 2): Takes the last 2 digits of the key.

  Example:

LastName	CustomerKey	Customer Code
Smith	201	Sm01
Lee	10542	Le42
Ng	7	Ng07

- **Validate Email Ends with '.com' and Contains '@'**

You need to create a calculated column that checks whether:

1. The EmailAddress contains @
2. And it ends with .com

If both conditions are true → return "Valid"
Else → return "Invalid"

DAX Formula:

dax

```
Valid Email? =
IF(
    FIND("@", DimCustomer[EmailAddress], 1, 0) > 0 &&
    RIGHT(DimCustomer[EmailAddress], 4) = ".com",
    "Valid",
    "Invalid"
)
```

Explanation:

- FIND("@", ..., 1, 0) > 0: Checks if @ exists
- RIGHT(..., 4) = ".com": Checks if email ends with .com
- IF(..., "Valid", "Invalid"): Returns validation result

Example:

Email Address	Result
john.doe@email.com	Valid
susan@mail.org	Invalid
invalidemail.com	Invalid

- **Extract Domain Name from EmailAddress**

ask: Extract Domain Name from EmailAddress

You want to extract the part after the @ symbol in the EmailAddress column (i.e., the domain).

DAX Formula:

dax

```
Email Domain =
MID(
    DimCustomer[EmailAddress],
    FIND("@", DimCustomer[EmailAddress]) + 1,
    LEN(DimCustomer[EmailAddress])
)
```

Explanation:

- FIND("@", ...): Finds the position of the @ symbol.
- + 1: Moves the start position to the character after @.
- MID(..., ..., LEN(...)): Extracts everything from that position to the end.

Example:

EmailAddress	Email Domain
john.smith@email.com	email.com
susan.miller@adventure.org	adventure.org

- **Mask Phone Number Except Last 4 Digits**

  You want to replace all characters in the phone number except the last 4 digits with asterisks (*), while preserving the total character length.

DAX Formula:

dax

```
Masked Phone =
REPT("*", LEN(DimCustomer[Phone]) - 4) &
RIGHT(DimCustomer[Phone], 4)
```

Explanation:

- LEN(DimCustomer[Phone]): Gets total length of the phone number.
- REPT("*", ... - 4): Repeats * for all but the last 4 characters.
- RIGHT(DimCustomer[Phone], 4): Keeps the last 4 digits visible.

Example:

Phone	Masked Phone
500-555-0161	********0161
1 (11) 555-1234	**********1234

This masks all characters, including symbols like -, (,).

- **Proper Casing of Last Name (simulate manually)**

  Since DAX doesn't have a built-in PROPER() function like Excel, we'll simulate proper casing manually:

- First letter uppercase
- Remaining letters lowercase

  DAX Formula:

  Proper LastName =

  UPPER(LEFT(DimCustomer[LastName], 1)) &

  LOWER(MID(DimCustomer[LastName], 2,
  LEN(DimCustomer[LastName]) - 1))

  Explanation:

- LEFT(..., 1): First character → UPPER()
- MID(..., 2, ...): Remaining characters → LOWER()
- Concatenate both parts

  Example:

LastName	Proper LastName
SMITH	Smith
o'CONNOR	O'connor
lee	Lee

- **Replace Multiple Spaces in EnglishOccupation with Single Space**

  You want to normalize the EnglishOccupation text by replacing all multiple consecutive spaces with a single space.

  Limitation in DAX

  DAX does not have built-in support for regular expressions, so it cannot directly detect "multiple spaces".

  Workaround in DAX: Nested SUBSTITUTE()

  We can simulate this by replacing double spaces repeatedly:

  dax

  CopyEdit

  Normalized Occupation =

  SUBSTITUTE(

  SUBSTITUTE(

  SUBSTITUTE(

  SUBSTITUTE(

  DimCustomer[EnglishOccupation],

  " ", " "),

  " ", " "),

  " ", " ")

This handles up to 4 consecutive spaces, but you can nest more SUBSTITUTE functions if needed.

Explanation:

- Each SUBSTITUTE(..., " ", " ") replaces double spaces with a single space.
- Multiple layers ensure any extra spaces beyond 2 are also reduced.

Example:

Original	Normalized
Senior Marketing Rep	Senior Marketing Rep
Production Technician	Production Technician

- **Generate Custom ID: Initials + Birth Year (e.g., JD_1985)**

  You want to create a Custom ID that consists of:

- First letter of FirstName
- First letter of LastName
- An underscore _
- Birth year from BirthDate

DAX Formula:

```
dax
Custom ID =
UPPER(LEFT(DimCustomer[FirstName], 1)) &
UPPER(LEFT(DimCustomer[LastName], 1)) &
"_" &
YEAR(DimCustomer[BirthDate])
```

Explanation:

- LEFT(..., 1): Gets first character of first and last names

- UPPER(...): Ensures initials are capitalized
- YEAR(...): Extracts the year from BirthDate
- & "_" &: Adds the underscore separator

Example:

FirstName	LastName	BirthDate	Custom ID
John	Doe	1985-04-12	JD_1985
Sarah	Kim	1992-09-30	SK_1992

- **Remove Hyphens and Convert Phone to Number**

  You want to:

1. Remove all hyphens (-) from the phone number
2. Convert the result into a numeric value

   DAX Formula:

   dax

   Phone as Number =

   VALUE(SUBSTITUTE(DimCustomer[Phone], "-", ""))

   Explanation:

- SUBSTITUTE(..., "-", ""): Removes all hyphens from the phone string
- VALUE(...): Converts the resulting text to a numeric data type (if valid)

  Example:

Phone	Result (Number)
500-555-0161	5005550161
1-800-123-4567	18001234567

Final DAX Formula (Clean & Convert Phone):

dax

```
Phone as Number =
VALUE(
   SUBSTITUTE(
      SUBSTITUTE(
         SUBSTITUTE(
            SUBSTITUTE(DimCustomer[Phone], "-", ""),
            "(", ""),
         ")", ""),
      " ", "")
)
```

What It Does:

- SUBSTITUTE(..., "-", "") → Removes hyphens
- SUBSTITUTE(..., "(", "") → Removes open parenthesis
- SUBSTITUTE(..., ")", "") → Removes close parenthesis
- SUBSTITUTE(..., " ", "") → Removes spaces
- VALUE(...) → Converts the cleaned text to a number

Example:

Original Phone	Cleaned Numeric
1 (11) 500-555-0161	1115005550161
500-555-0161	5005550161
(123) 456 7890	1234567890

- **Create a measure or calculated column that categorizes customers into segments using both EnglishEducation and YearlyIncome.**
  If the education is "Graduate Degree" and income > 90000 → "Elite"
  If education is "Bachelors" and income between 60000–90000 → "Professional"
  If education is "High School" → "Basic"
  Otherwise → "Other"

  Logic:

- If EnglishEducation = "Graduate Degree" and YearlyIncome > 90000 → "Elite"
- If EnglishEducation = "Bachelors" and YearlyIncome between 60000–90000 → "Professional"
- If EnglishEducation = "High School" → "Basic"
- Else → "Other"

  Recommended: Calculated Column

  Use this if you want to assign a category per row in the DimCustomer table:

  dax

```
Customer Segment =
SWITCH(
    TRUE(),
    DimCustomer[EnglishEducation] = "Graduate Degree" &&
DimCustomer[YearlyIncome] > 90000, "Elite",
    DimCustomer[EnglishEducation] = "Bachelors" &&
DimCustomer[YearlyIncome] >= 60000 && DimCustomer[YearlyIncome]
<= 90000, "Professional",
    DimCustomer[EnglishEducation] = "High School", "Basic",
    "Other"
)
```

  Why SWITCH(TRUE(), ...)?

- Allows you to write multi-condition IF statements in a cleaner, readable format.
- Each row is evaluated in order from top to bottom.

Example Output:

EnglishEducation	YearlyIncome	Customer Segment
Graduate Degree	95000	Elite
Bachelors	65000	Professional
High School	42000	Basic
Partial College	71000	Other

- **Create a measure that returns:**
  Total Customers if no selection
  Customer count for selected Gender
  If more than one gender is selected, return "Multiple Values Selected"
- Total customers if no gender is selected
- Customer count for selected gender
- "Multiple Values Selected" if more than one gender is selected

DAX Measure:

dax

```
Customer Gender Filtered =
VAR SelectedGenders = VALUES(DimCustomer[Gender])
VAR GenderCount = COUNTROWS(SelectedGenders)

RETURN
SWITCH(
    TRUE(),
```

```
    GenderCount = 0, CALCULATE(COUNTROWS(DimCustomer)),  -- No filter
    GenderCount = 1, CALCULATE(COUNTROWS(DimCustomer)),  -- One gender selected
    "Multiple Values Selected"                           -- More than one selected
)
```

Explanation:

- VALUES(DimCustomer[Gender]): Returns a list of selected genders in the filter context.
- COUNTROWS(...): Tells how many genders are selected.
- SWITCH(TRUE(), ...): Handles the three possible scenarios:
    1. 0 genders selected → return total
    2. 1 gender selected → return filtered count
    3. More than 1 selected → return a message

Note:

The final result must always return either text or number, so to handle text + number hybrid nicely for reporting, we can format everything as text:

Optional Version (Always Returns Text):

dax

```
Customer Gender Filtered (Text) =
VAR SelectedGenders = VALUES(DimCustomer[Gender])
VAR GenderCount = COUNTROWS(SelectedGenders)
VAR CustomerCount = CALCULATE(COUNTROWS(DimCustomer))

RETURN
SWITCH(
    TRUE(),
```

```
    GenderCount = 0, "Total Customers: " & CustomerCount,
    GenderCount = 1, "Selected Gender Count: " & CustomerCount,
    "Multiple Values Selected"
)
```

Output Examples:

Selected Gender	Output
None	"Total Customers: 18484"
Male	"Selected Gender Count: 9821"
Female + Male	"Multiple Values Selected"