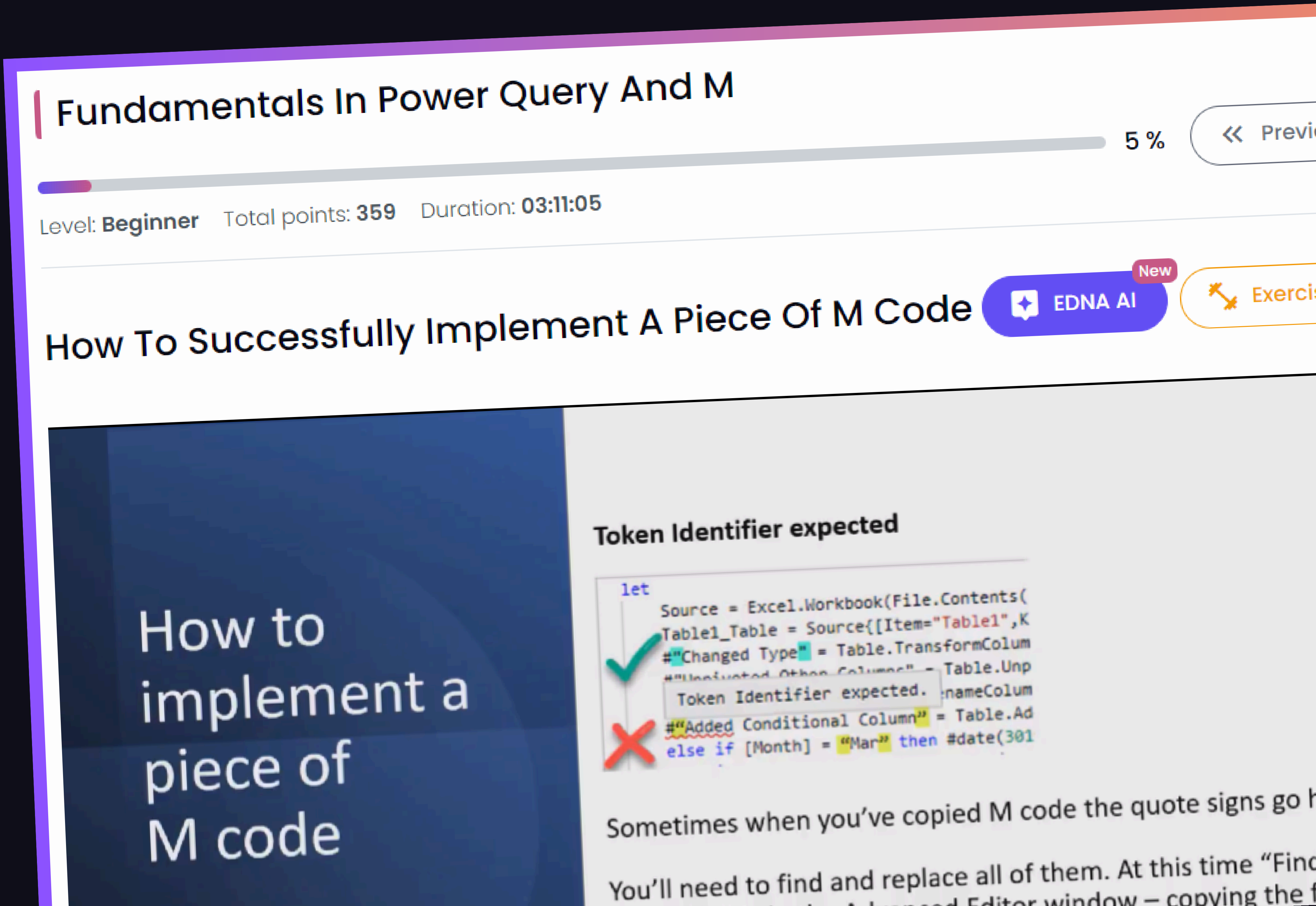
ENTERPRISEDNA.CO

BROUGHT TO YOU BY ENTERPRISE DNA

Power Query/M

The Fundamentals

- a detailed guide



Fundamentals In Power Query And M

5 %

Previ

Level: **Beginner** Total points: **359** Duration: **03:11:05**

How To Successfully Implement A Piece Of M Code

Token Identifier expected

```
let
    Source = Excel.Workbook(File.Contents(
    Table1_Table = Source{[Item="Table1",K
    #"Changed Type" = Table.TransformColum
    #"Unpivoted Other Columns" = Table.Unp
     Token Identifier expected.        nameColum
    #“Added Conditional Column” = Table.Ad
    else if [Month] = “Mar” then #date(301
```

Sometimes when you’ve copied M code the quote signs go

You’ll need to find and replace all of them. At this time “Find

Advanced Editor window – copying the

21 POWER QUERY TOPICS

- Introduction to Power Query and M
- Overview of Power Query in Power BI
- Understanding the Power Query Editor Interface
- Configuration Options for Power Query Editor
- Using the M Formula Language for Data Transformation
- Navigating the Query Editor Ribbons and Menus
- Essential Functions in M Language
- Error Handling Techniques in Power Query
- Writing and Modifying M Code
- Custom Functions in M
- Optimization and Performance Enhancement
- Creating and Using Parameters in Queries
- Data Types in M
- Operators in M
- Understanding and Applying Context in M
- Advanced Data Manipulation with M
- Managing your Data Model in Power Query
- Managing Relationships in Power Query
- Best Practices for Writing Efficient M Code
- Debugging and Validating M Formulas
- Handling Totals and Virtual Tables in Power Query

1

Introduction to Power Query/M

Power Query is a data connection technology that enables you to discover, connect, combine, and refine data sources to meet your analysis needs. Features include comprehensive data manipulation capabilities, allowing for easy data cleaning, reshaping, and transformation.

Within Power BI, Power Query acts as the data connection interface. Here, you can import data from a vast array of sources, then transform and load that data into the Power BI model. Power Query is integrated into Power BI Desktop and is accessible via the Home tab under the Data section.

Key Features

- **Integration**: Seamlessly integrated with Excel and Power BI, providing a consistent experience across multiple platforms.

- **Connectivity**: Connects to hundreds of data sources including files, databases, web services, and more.

- **Transformations**: Offers a wide range of data transformation functions that are accessible through a user-friendly interface.

DATA MENTOR ENTERPRISE DNA

2

Overview of Power Query in Power BI

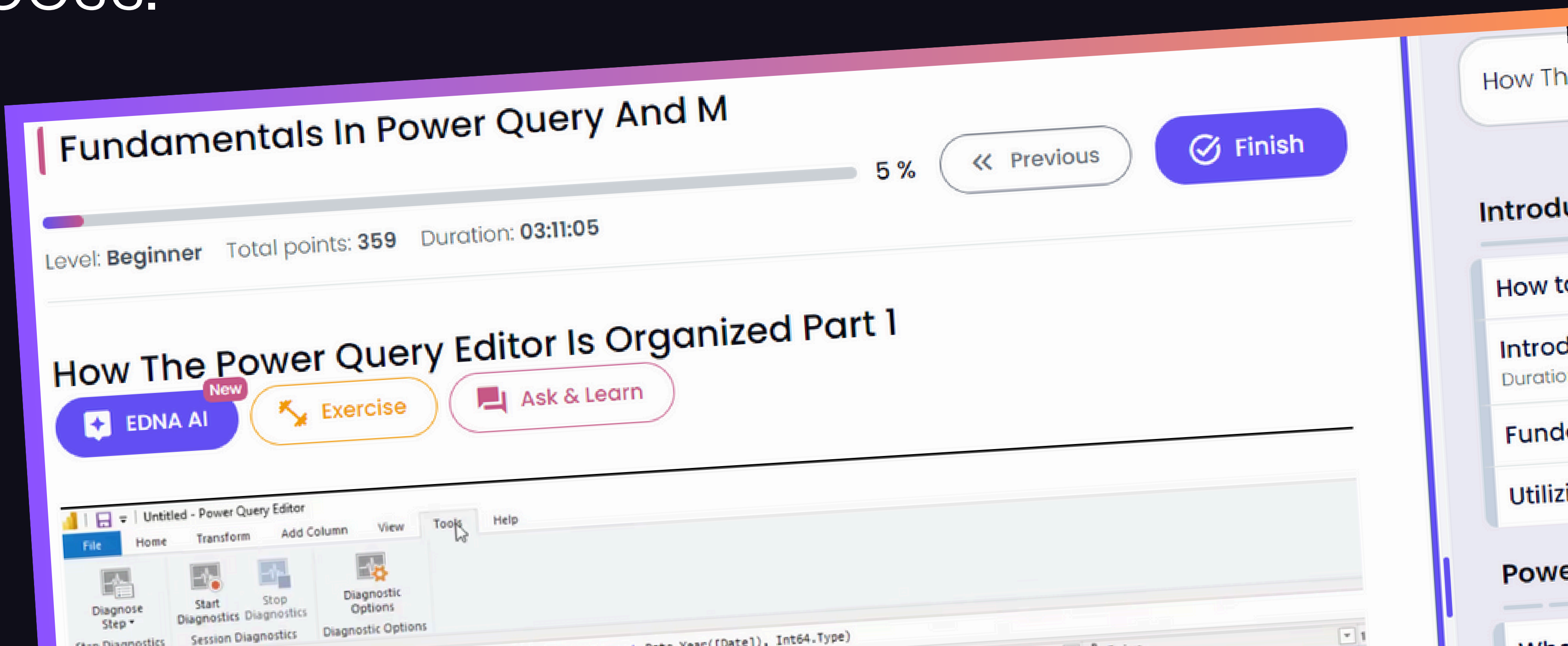
Power Query in Power BI is essential for data ingestion and preprocessing. It enables users to cleanse data, merge datasets, and transform them for analysis and visualization.

- Combining data from different sources (e.g., SQL databases, Excel files, web pages).
- Cleansing data by removing duplicates, filtering rows, or converting data types.
- Enriching data by adding new calculated columns or merging tables.

3

Understanding the Power Query Editor Interface

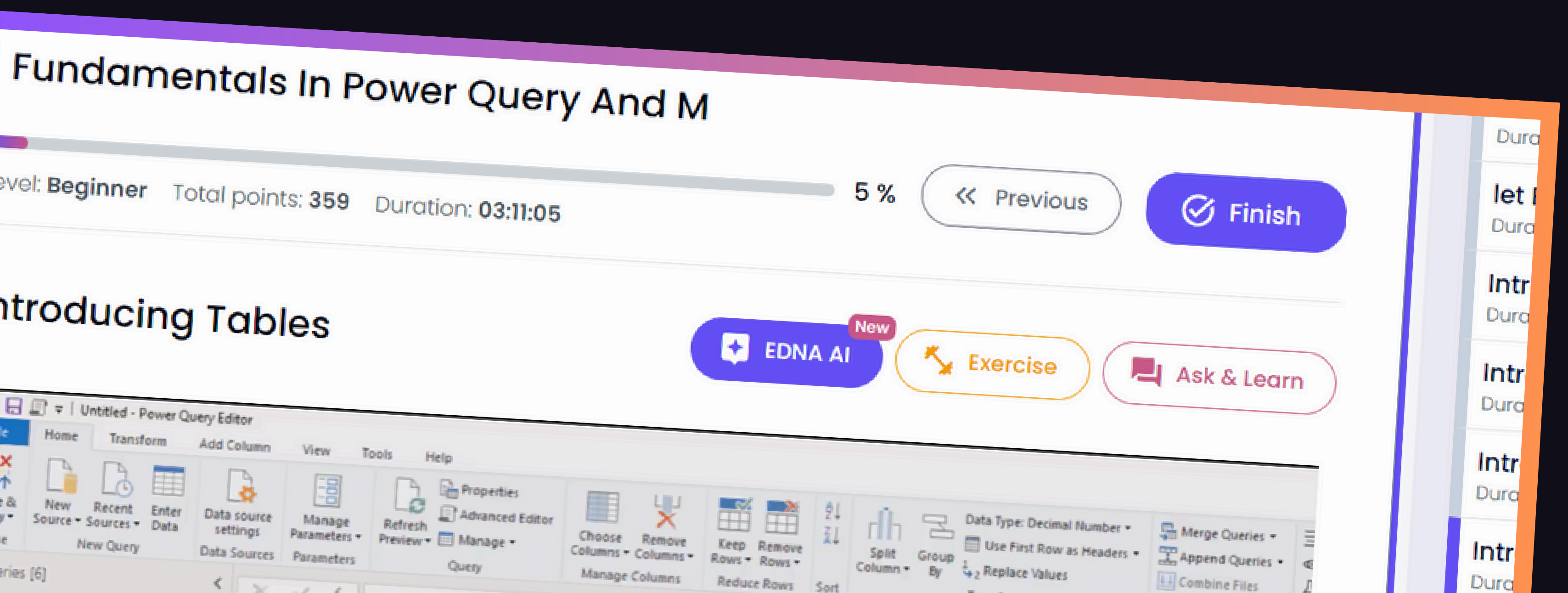
- Ribbon: Contains tools for adding new sources, managing queries, and applying transformations.
- Query List: Displays all queries involved in the report, allowing easy navigation and management.
- Formula Bar: Shows the M code for any transformations applied, providing insight into the data manipulation process.



4

Configuration Options for Power Query Editor

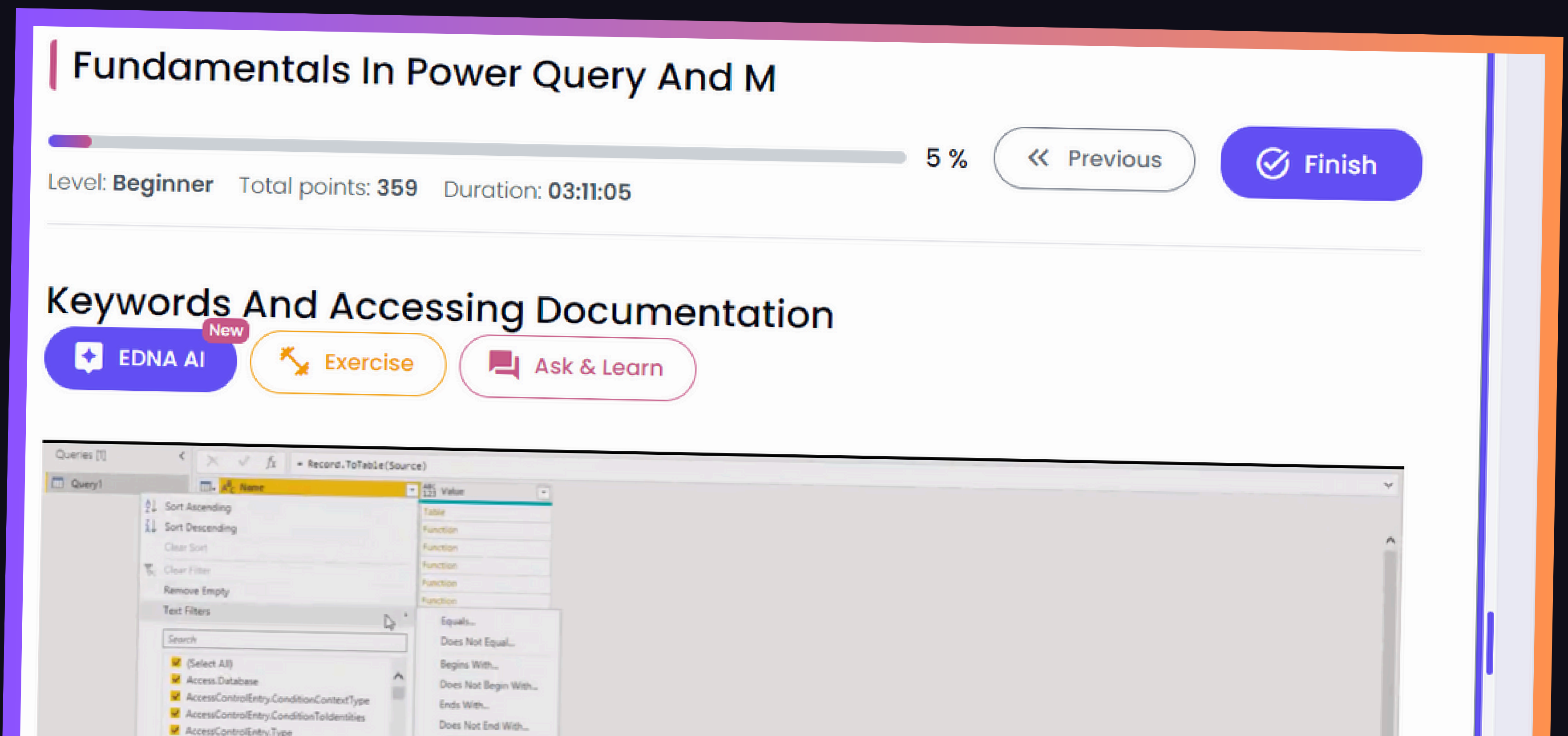
- Data Load: Configure default settings for loading data into the model, like disabling load or adjusting load behavior for performance optimization.
- Query Options: Set parameters such as timeout durations for data sources or enable/disable background data preview.



5

Using the M Formula Language for Data Transformation

- M Language: A powerful, case-sensitive language used in Power Query for data transformation tasks.
- Functionality: Supports a wide array of functions like text operations, date manipulations, conditional logic, and loop structures.


Fundamentals In Power Query And M

5 % Previous Finish

Level: **Beginner** Total points: **359** Duration: **03:11:05**

Keywords And Accessing Documentation

Navigating the Query Editor Ribbons and Menus

The Query Editor in Power BI features several ribbons and menus designed to facilitate efficient data manipulation and transformation.

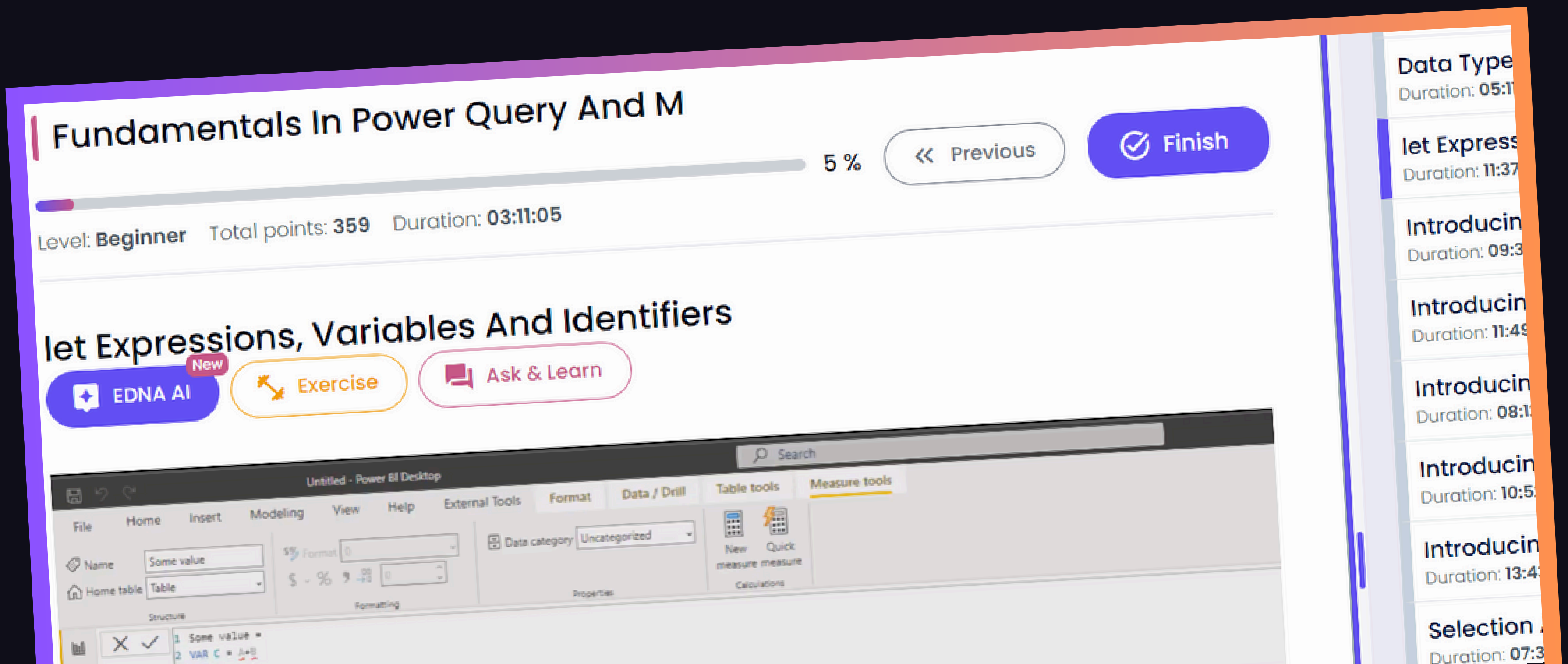
- Home Ribbon: Quick access to essential functions like New Source, Combine, and Transform Data.
- Transform Ribbon: Dedicated to more detailed data transformations such as grouping, pivoting, and advanced calculations.

7

Essential Functions in M Language

Explore the foundational functions in M, crucial for data transformations and manipulations within Power Query.

- Textual: Functions for string manipulation like Text.Combine, Text.Proper, and Text.Split.
- Numerical: Operations like Number.Round, Number.Abs, and arithmetic functions.
- Logical: Conditional logic functions including If, And, Or.



8

Error Handling Techniques in Power Query

Errors in Power Query typically arise from data inconsistencies, incorrect type operations, or issues in source data.

Common errors include:

- Type mismatch: Occurs when an operation or function is applied to an incompatible data type.
- Missing data: Errors due to NULL or blank entries that are not handled correctly.
- Syntax errors: Mistakes in the M code such as misspelled functions, incorrect arguments, or improper formatting.

PRO TIPS

- **Data Cleansing**: Before performing transformations, cleanse the data to reduce potential errors. For example, replace or remove null values, trim text, or standardize date formats.

- **Logging Errors**: Create a separate query or column to log error details, which can help in debugging and improving the data preparation process.

- **Preventative Checks**: Use data type checks and conditional statements to prevent operations on incompatible types. This proactive approach minimizes runtime errors.

DATA MENTOR ENTERPRISE DNA

9

Writing and Modifying M Code

M code, or the M language, is the scripting language used in Power Query. It's a functional, case-sensitive language designed specifically for data manipulation and transformation.

- Expressions: The building blocks of M code, which perform actions and return results. Everything in M, from simple calculations to complex data transformations, is handled through expressions.
- Functions: Predefined or custom routines that perform specific tasks and can be reused across your queries.
- Queries: Named sequences of steps, written in M, that transform data. Each step in a query is typically an expression involving one or more functions.

PRO TIPS

- **Parameterization**: Make your queries more dynamic and reusable by introducing parameters. Parameters can be used to control inputs like file paths, filter conditions, and other variables without hardcoding them into your queries.

- **Custom Functions**: Create custom functions to handle repetitive tasks or complex logic. This not only makes your main queries cleaner but also promotes code reuse.

- **Refactoring**: As your understanding of M grows, revisit old scripts to refactor them. Simplifying expressions, consolidating steps, and improving performance by minimizing data duplication are key goals in refactoring.

DATA MENTOR ENTERPRISE DNA

10

Custom Functions in M

- A function is defined using the let…in syntax, where let allows you to define the function's internal logic, and in specifies the output of the function.
- Functions can accept parameters of any data type, including numbers, text, tables, and records. You can also specify the data type of the return value for clarity and error handling.
- Functions only have access to the variables and parameters defined within their scope. They can be stored as part of a query or as separate query items within a Power BI model.

11

Optimization and Performance Enhancement

Identifying bottlenecks is the first step toward optimization. Common areas where performance issues may arise in Power Query include:

- Large Data Volumes: Handling substantial amounts of data can slow down data refresh times.
- Complex Calculations: Extensive use of complex calculations or iterating over large datasets can degrade performance.
- Inefficient Queries: Poorly structured M queries that don’t take advantage of query folding can lead to unnecessary data loading and transformation.

12

Creating and Using Parameters in Queries

- Flexibility: Easily adapt your queries to new requirements without modifying the underlying M code.
- User Interactivity: Allow end-users to interact with data models and reports by setting parameters directly in Power BI reports.
- Enhanced Reusability: Use the same parameterized query to process different datasets or apply different filters by simply changing parameter values.

PRO TIPS

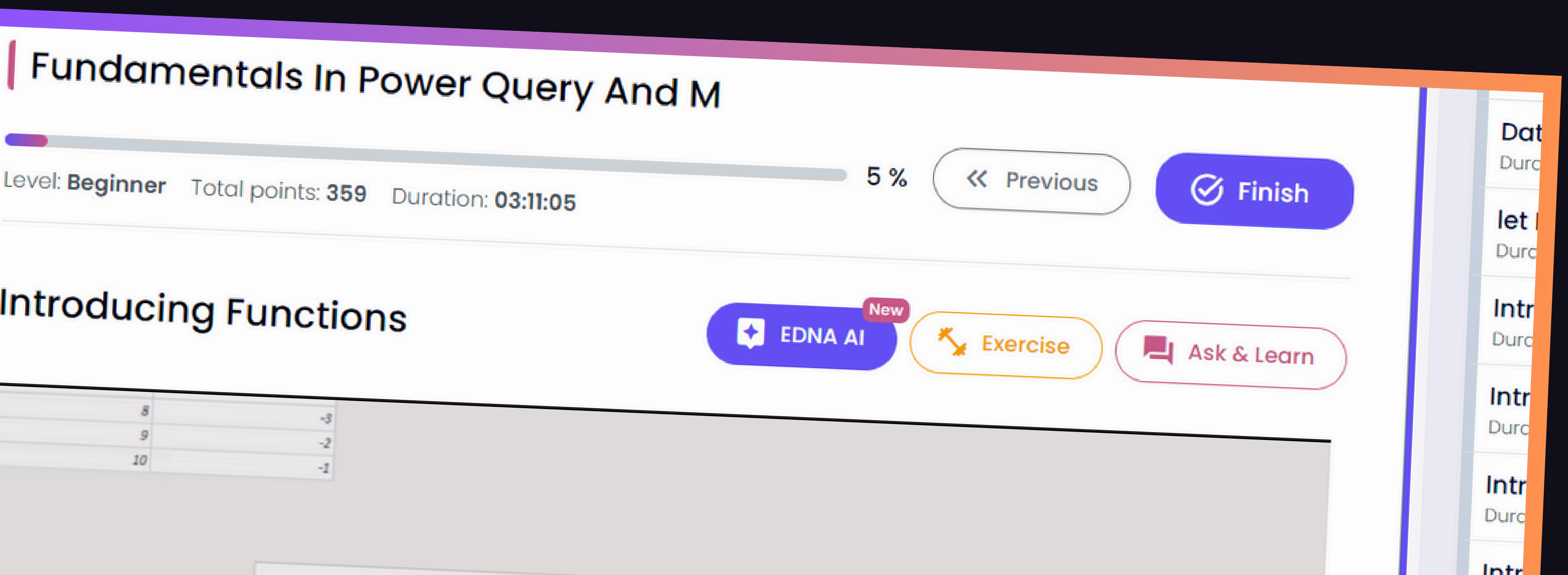
- **Clear Naming Conventions**: Use meaningful names for parameters to ensure they are understandable and maintainable.
- **Validation**: Provide a list of possible values for parameters to prevent errors due to incorrect inputs.
- **Documentation**: Document the purpose and expected values for parameters, especially when they are exposed to end-users in Power BI reports.

13

Data Types in M

M provides a rich set of data types that are foundational for manipulating data structures. These types include:

- Primitive Types: These are the basic types that include numbers, text, logical (boolean), null, and binary.
- Structured Types: These include lists, records, and tables, which are collections of values.
- Date and Time Types: Dedicated types for handling dates, times, datetime values, and durations.



14

Operators in M

- Arithmetic Operators
  Used for basic math operations: +, -, *, /, ^ (exponentiation).
- Comparison Operators
  Compare two values, returning a boolean: =, <>, <, >, <=, >=.
- Text Operators
  Concatenate text strings: &.
- Logical Operators
  Combine boolean expressions: and, or, not.
- List, Record, and Table Operators
  Access and manipulate collections: {} for lists, [] for records.
- Advanced Functional Operators
  Simplify function expressions: each for applying functions to each element, => for defining lambda functions.

15

Understanding and Applying Context in M

In M, the formula language for Power Query, context is key for data manipulation. Row context applies to calculations within a row, while query context affects how queries interact and how parameters modify data flow.

Mastering these contexts boosts the flexibility and interactivity of Power BI reports, ensuring that queries are robust and respond dynamically to user inputs and data changes. Clear referencing and consistent testing are essential for effective context management.

16

Advanced Data Manipulation with M

Advanced data manipulation in M includes essential techniques for handling complex transformations efficiently.

Functions like `Text.Split`, `Text.Combine`, `Table.NestedJoin`, and `Table.Combine` enable sophisticated text processing and data merging.

Robust error handling with `Try...Otherwise` and dynamic date manipulations like `Date.AddMonths` ensure precision and resilience in data operations, streamlining analytics workflows.

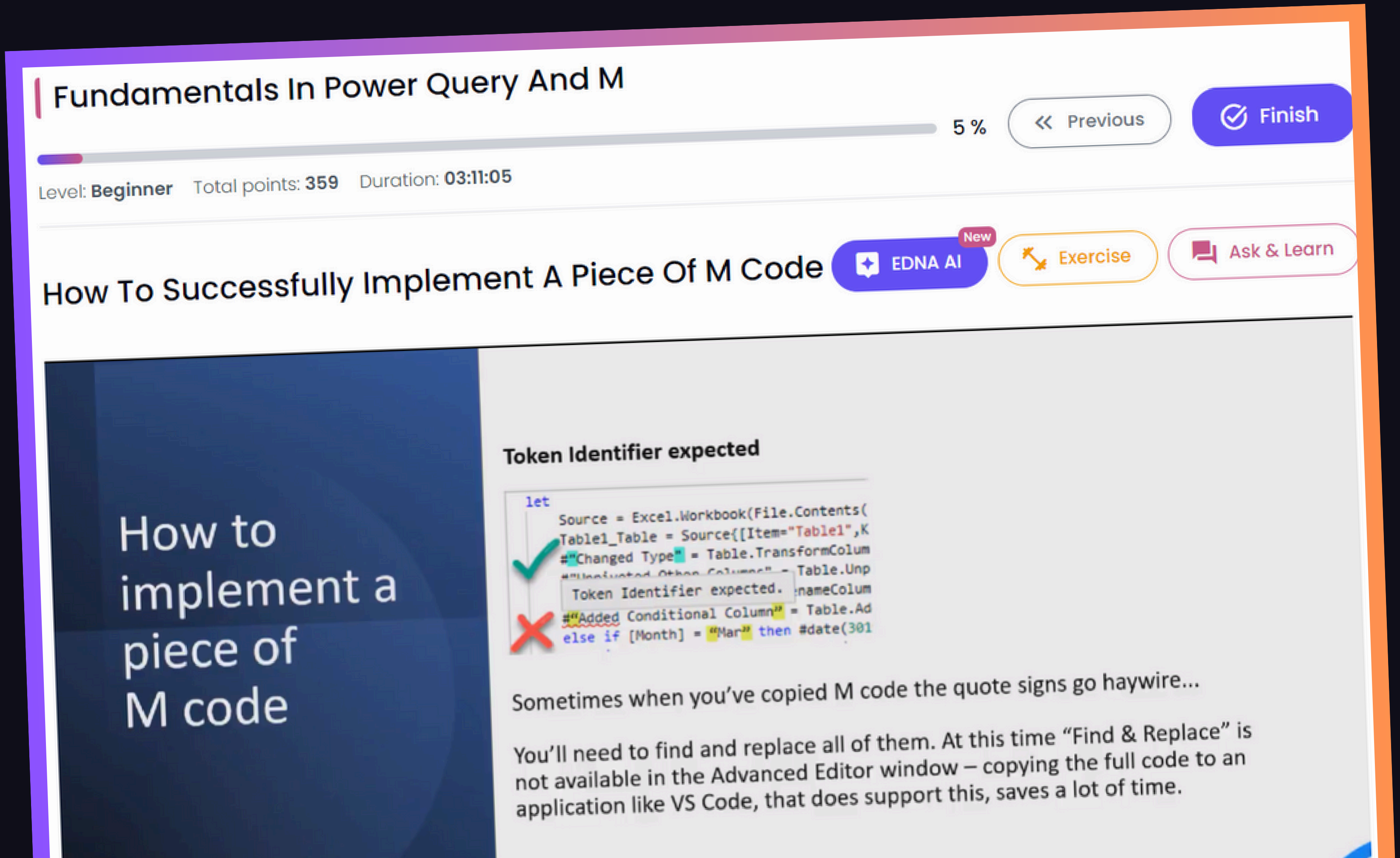
Managing your Data Model

- In Power Query, structuring data involves transforming and organizing raw data into a more usable format.
- You might often need to create new tables within Power Query to better organize your data or to create summary tables. Using functions like Table.Group, Table.Summarize, and others can help in aggregating data and restructuring it as needed.
- Essential to data modeling is the process of cleaning data

18

Managing Relationships

Managing relationships in Power Query within Power BI involves linking different data tables through common columns to ensure that the data model reflects the inherent connections between data entities. This is crucial for creating effective and accurate reports and analyses.



How To Successfully Implement A Piece Of M Code

Token Identifier expected

```
let
    Source = Excel.Workbook(File.Contents(
    Table1_Table = Source{[Item="Table1",K
    #"Changed Type" = Table.TransformColum
    #"Unpivoted Other Columns" = Table.Unp
     Token Identifier expected.           nameColum
    #“Added Conditional Column” = Table.Ad
    else if [Month] = “Mar” then #date(301
```

Sometimes when you've copied M code the quote signs go haywire...

You'll need to find and replace all of them. At this time "Find & Replace" is not available in the Advanced Editor window – copying the full code to an application like VS Code, that does support this, saves a lot of time.

Best Practices for Writing Efficient M Code

- Minimize Data Loading: Load only necessary data by importing required columns and filtering rows early, reducing memory usage and speeding up data processing.
- Promote Query Folding: Leverage query folding to push data transformations back to the data source, especially with SQL databases. This reduces Power BI's workload by utilizing the database server's processing power.
- Use Native Functions: Prefer built-in Power Query functions over custom M code for common data transformations, as they are more optimized and likely to support query folding.

- Optimize Data Type Management: Be cautious with data type conversions. Maintain consistent data types from initial import to avoid unnecessary processing overhead during transformations.
- Utilize Buffering Sparingly: Use Table.Buffer and List.Buffer judiciously to cache data during transformations. Buffering can enhance performance but increases memory consumption.
- Maintain Code Clarity and Simplicity: Write clear, well-documented M code. Use comments for complex logic, maintain a consistent coding style, and break down complicated transformations for better readability and maintainability.
- Profile and Optimize Queries: Regularly test and profile your M code with tools like Query Diagnostics in Power BI to identify and address performance bottlenecks.

20

Debugging and Validating M Formulas

- Understand the Data: Before diving into debugging, ensure you have a thorough understanding of the data structure, types, and expected outcomes.
- Use the Advanced Editor: Use this tool to examine the entire query script, which can help you identify syntax errors or logic flaws.
- Step-by-Step Execution: This approach helps isolate the step where the error occurs.
- Check Data Types: Ensure that each operation’s data types are compatible.
- Use Error Handling Functions: M includes functions designed to handle errors within your data transformations.

- Validate Results with Test Data: This is particularly useful for complex transformations where the logic needs to be verified against specific scenarios.
- Watch for Performance Issues: Look for opportunities to promote query folding, minimize the use of row-wise operations, and leverage buffering wisely.
- Use Comments and Documentation: Adding comments within your M code helps document what each part of the query is intended to do.
- Leverage Community Resources: If you encounter a particularly challenging issue, consider leveraging resources such as the Microsoft Power BI Community forums, Stack Overflow, or other online resources where many common (and uncommon) issues have been discussed and resolved.
- Continuous Monitoring and Testing: Automated testing and monitoring can be set up for critical reports to ensure ongoing accuracy.

21

Handling Totals and Virtual Tables in Power Query

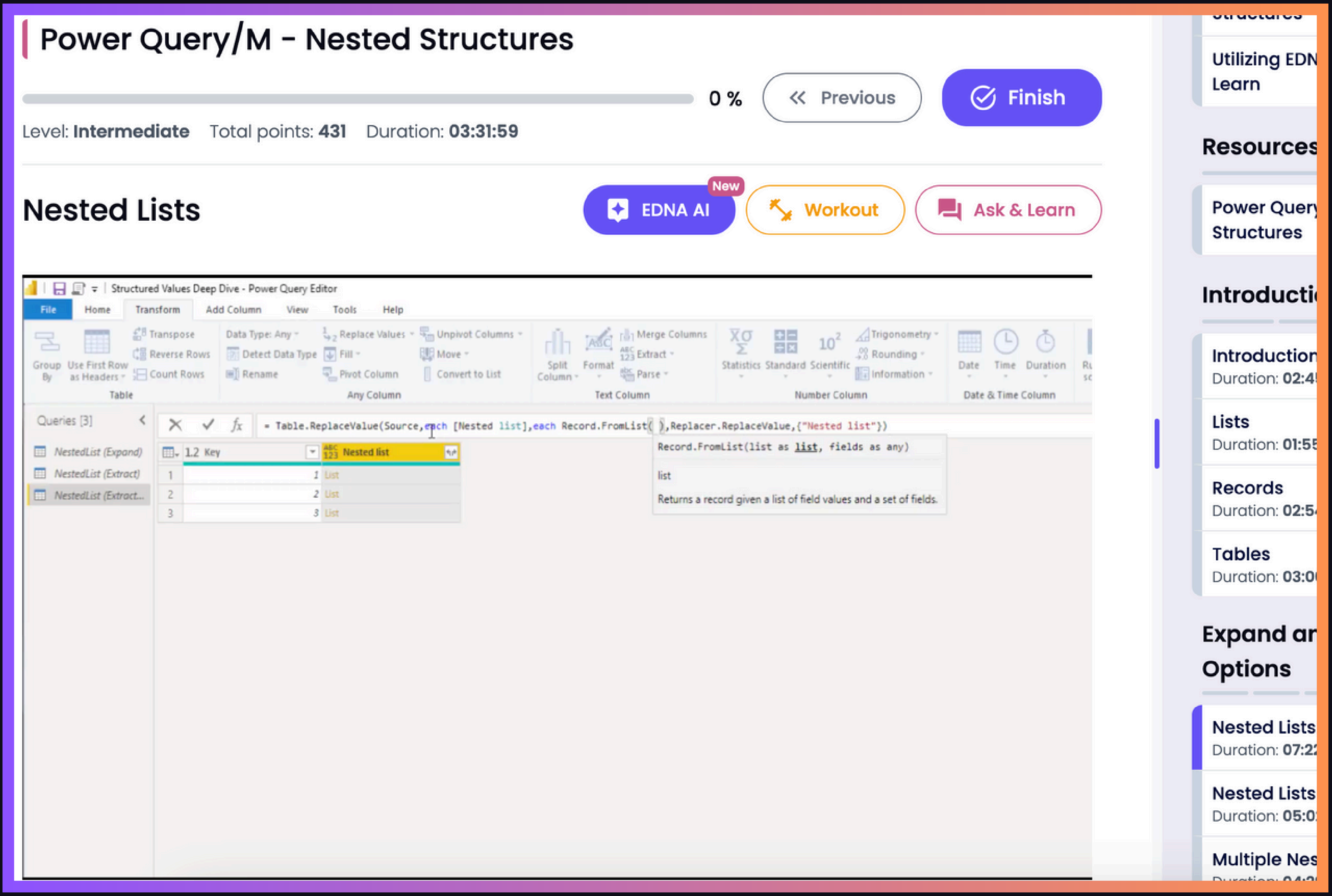
- Handling totals and managing virtual tables are essential in Power Query and DAX for creating comprehensive, dynamic reports in Power BI.
- Totals are used in Power BI to provide aggregated summaries of data, such as sums, averages, counts, and more. They are crucial for giving users a high-level view of key metrics.
- Virtual tables are temporary tables created on the fly during the execution of a DAX query. They are used to hold intermediate results, perform calculations, or reshape data in ways that are not directly supported by the underlying data model.

PRO TIPS

- Clear Definitions: Ensure that all measures involving totals and virtual tables are clearly defined and documented, particularly explaining their business logic and context.
- Performance Monitoring: Regularly review the performance impact of virtual tables and complex total calculations, especially in larger datasets or more complex models.
- Testing and Validation: Continuously test and validate the calculations against expected results to ensure accuracy and reliability, particularly when deploying new or updated reports.

ENTERPRISE DNA

Check out 'Power Query/M – Nested Structures' to learn more



Power Query/M - Nested Structures

0 % « Previous Finish

Level: **Intermediate** Total points: **431** Duration: **03:31:59**

Nested Lists

EDNA AI New Workout Ask & Learn

Structured Values Deep Dive - Power Query Editor

File Home Transform Add Column View Tools Help

Queries [3]

NestedList (Expand)
NestedList (Extract)
NestedList (Extract...

```
= Table.ReplaceValue(Source,each [Nested list],each Record.FromList( ),Replacer.ReplaceValue,{"Nested list"})
```

Record.FromList(list as list, fields as any)

list

Returns a record given a list of field values and a set of fields.

	1.2 Key	Nested list
1	1	List
2	2	List
3	3	List

Utilizing EDN
Learn

Resources

Power Query
Structures

Introducti

Introduction
Duration: 02:4

Lists
Duration: 01:55

Records
Duration: 02:5

Tables
Duration: 03:0

Expand an
Options

Nested Lists
Duration: 07:2

Nested Lists
Duration: 05:0

Multiple Nes

Course now available on EDNA Learn

*This resource content was created from this comprehensive course within Enterprise DNA's learning platform EDNA Learn

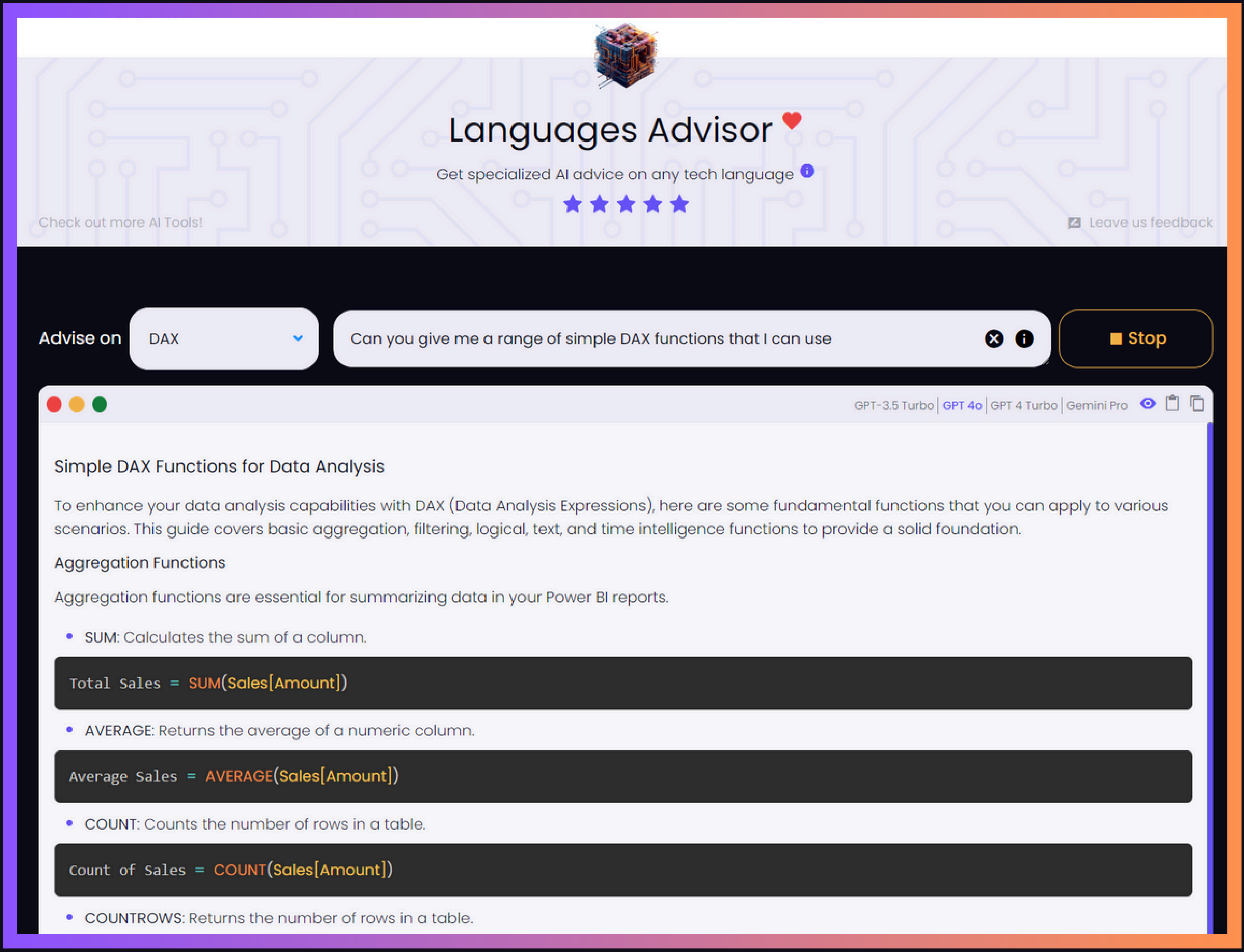
This is fancy QR code. Give it a try.

ENTERPRISE DNA

For an further deep dives into Power Query or M code topics check out

Languages Advisor

DATA MENTOR ENTERPRISE DNA



Languages Advisor

Get specialized AI advice on any tech language

Check out more AI Tools!

Leave us feedback

Advise on DAX

Can you give me a range of simple DAX functions that I can use

Stop

GPT-3.5 Turbo | GPT 4o | GPT 4 Turbo | Gemini Pro

Simple DAX Functions for Data Analysis

To enhance your data analysis capabilities with DAX (Data Analysis Expressions), here are some fundamental functions that you can apply to various scenarios. This guide covers basic aggregation, filtering, logical, text, and time intelligence functions to provide a solid foundation.

Aggregation Functions

Aggregation functions are essential for summarizing data in your Power BI reports.

- SUM: Calculates the sum of a column.

```
Total Sales = SUM(Sales[Amount])
```

- AVERAGE: Returns the average of a numeric column.

```
Average Sales = AVERAGE(Sales[Amount])
```

- COUNT: Counts the number of rows in a table.

```
Count of Sales = COUNT(Sales[Amount])
```

- COUNTROWS: Returns the number of rows in a table.

Get involved in the conversation

Linkedin

X (Twitter)