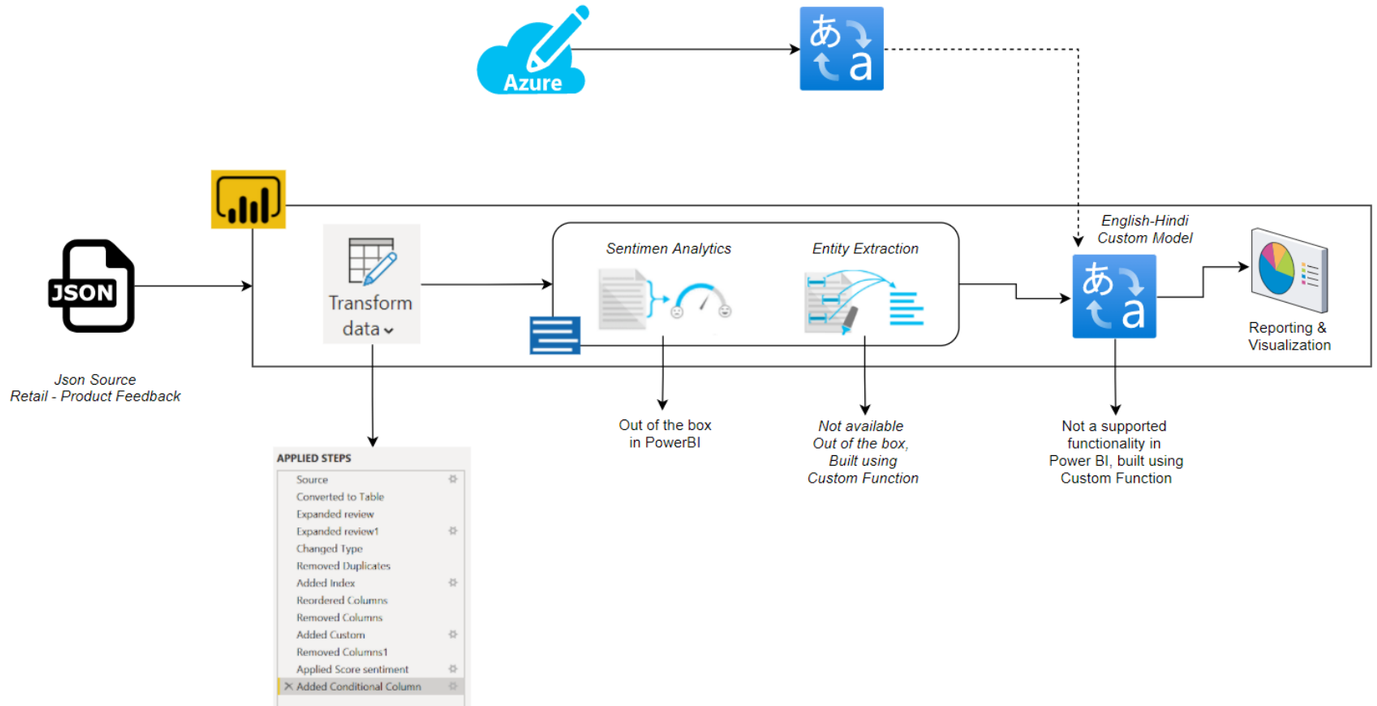
Document Name	HOL – Localisation of Reports using Text Analytics & Translator
Author	Shiva S Tomar & Anupreet Kaur
Reviewer	
Executive Summary	This workshop is an end-to-end simulation of processing text data, applying text analytics and translating the results to a local language to build powerful PowerBI reports that can cater to the local audience. To accomplish this workshop, we will be using Azure services like Cognitive Services (Translation and Text Analytics services) and PowerBI.
Purpose	The end-to-end scenario covered in this workshop will help you gain level 400 exposure on various Azure services, based on the knowledge you have gained by working on the previous workshops on individual cognitive services. Currently, PowerBI does not provide out of the box modules to perform Entity Extraction and Text translation capabilities. Once you complete this lab, you will be the **Jedi Master** on the Azure services and should be able to ***D**emo, **D**evelop and **D**eploy* a POC or work in a production environment involving these Azure services. The important thing to note here is that you don't need to refer any other documents to complete this workshop.
Intent of Guide	This workshop is designed to help you integrate and automate your solutions using Azure services that lie in the Data & AI landscape. Also, we understand that Data folks come from a SQL or Python background and have limited exposure to application languages like .NET, Java etc. Hence, we have designed and developed this use case using no-code low-code and GUI driven services. It also covers the Concepts, How-to and best practices about the service.

Use case brief: Localised Translated Text Analytics & Reporting

In today's world we interact with people from different native backgrounds. The corporate world does consider English as the primary language, however, there are a lot of folks out there who are more comfortable working in their native languages and can better understand the data and results when they are presented to them in their local language.
For example, India has a very diverse culture with over 30 major languages and a lot many local languages. Each state government prefers to deal with data and reports in their local language that they are most comfortable with. This is a common ask across the globe when you work with people from different backgrounds.
Thus, we have built a use case that takes unstructured data as input [free text], transforms it to a structured format [using Text Analytics & Natural Language Processing], translates into a local language of your choice [in our case Hindi] and uses it to create reports to cater to the users as required.
Alternatively, you can also create reports in any of the local languages that are supported by Azure Translator service, which includes over 12 Indian languages and over 90 languages globally.

Diagram: Functional Architecture



In this scenario, we will be leveraging customer reviews' data for a retail scenario. These reviews are present in a JSON file. Majority of our tasks will be performed in PowerBI. A summary of the tasks being performed is as follows :

1. Ingest & process the JSON file by appying certain transformations
2. Extract 'sentiment score' for the customer reviews by using out-of-the box Sentiment Analytics capability in PowerBI
3. Extract 'entities' from the customer reviews by writing custom queries and API calls since this capability is not available out-of-the-box in PowerBI yet
4. Translate the customer reviews and entities in a local language (Hindi) by writing custom queries and API calls since Translator service is not integrated with PowerBI yet. To get better results, we will be using the custom model we created in Translator Services' workshop.
5. Create reports that can cater to audience that wants to view the content in a local languge that they are more familiar with

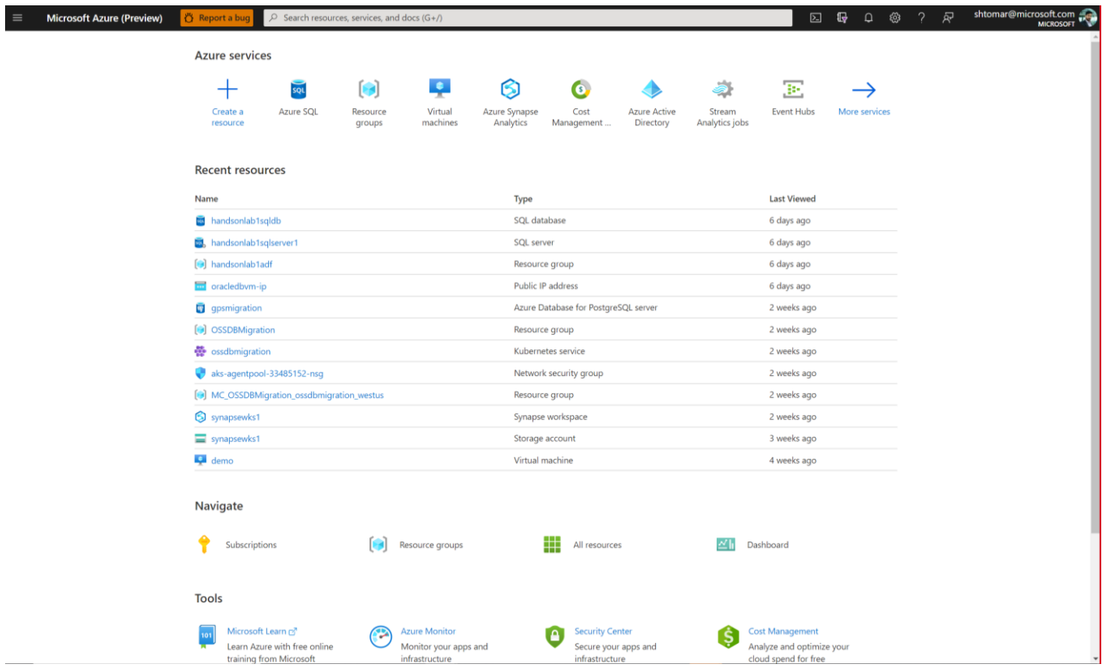
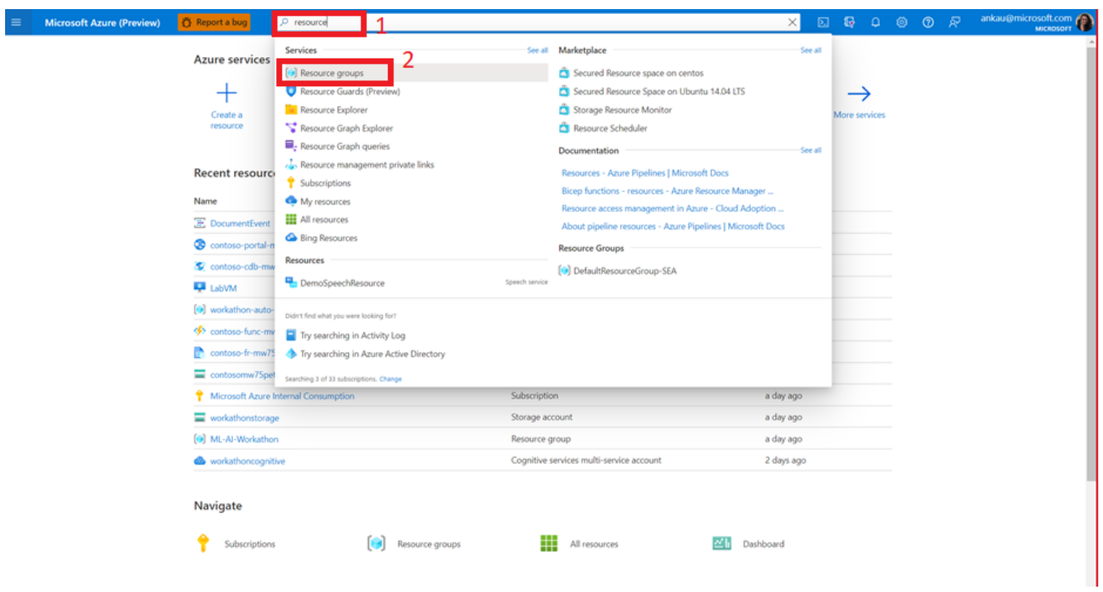
For this workshop, we are performing the Cognitive Services task from PowerBI. This is to help you learn how you can create custom queries and make API calls from PowerBI. In a production scenario, as a best practice, you will apply the cognitive services tasks on the data prior to ingesting it in PowerBI.

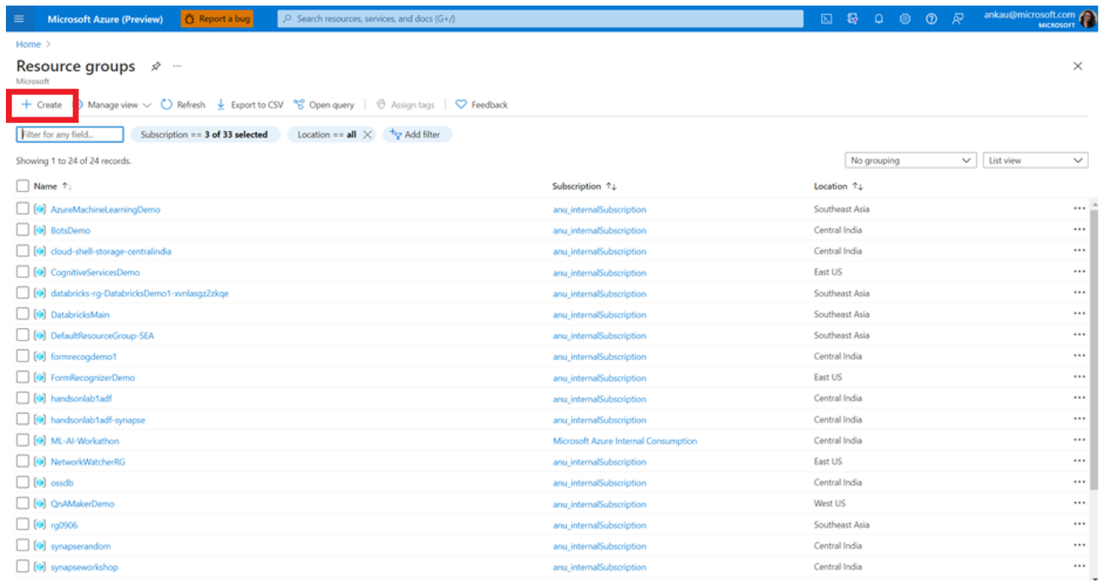
Step by step hands on guide to go from *Zero to Hero*

Pre-requisites

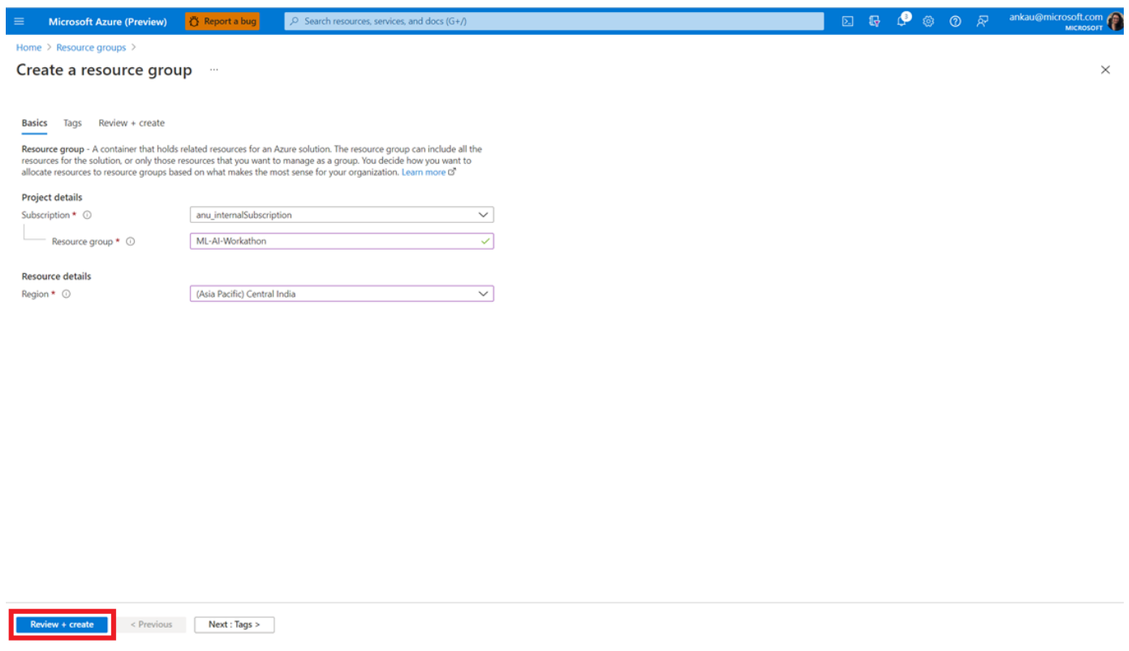
- An active Azure Account
  - You can use your current Azure Subscription or get started by creating a free trial account (https://azure.microsoft.com/en-in/free)
- Completed the workshops on Translation & Text Analytics and understand the workflow for Text Translation API and Text Analytics API.
- Download the data for customer reviews from Data Folder.

Let's get started!

Screenshots	Steps & Significance
	Sign into your Azure Portal.
	## Create a Resource Group Follow steps 1 & 2 to create a resource group. You can skip this step if you already have a Resource Group in place.



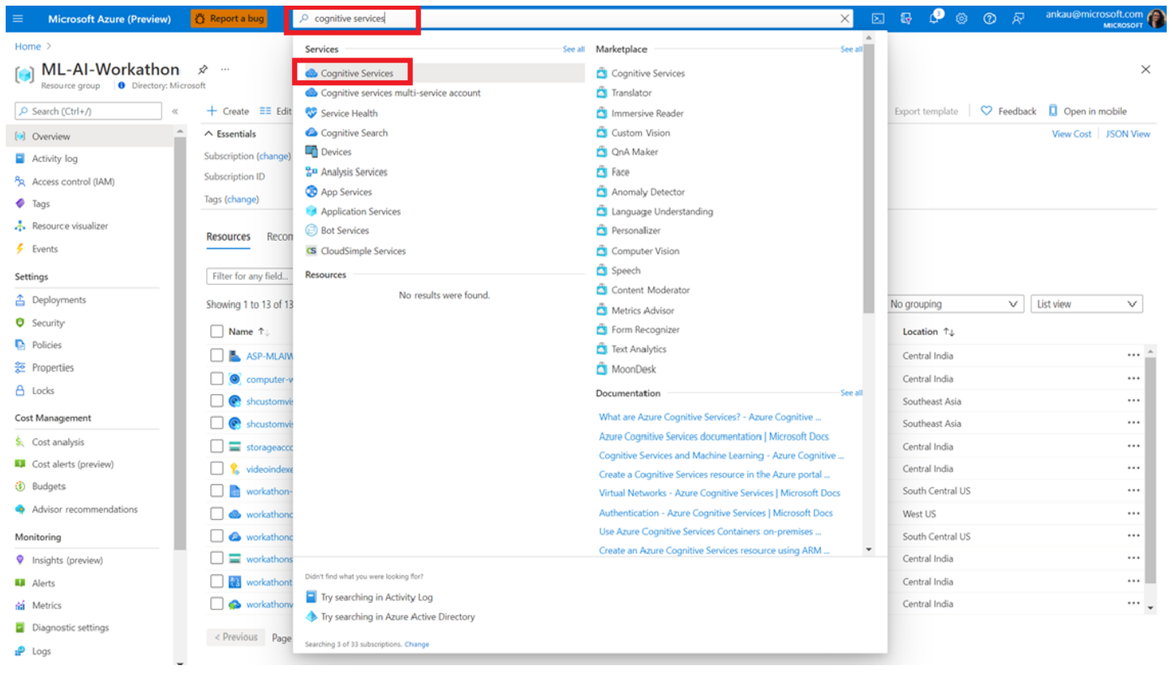
Click create to create a new resource group.



Enter the details –

1. Subscription : Azure subscription in which you want to deploy the resource group
2. Resource Group : Name of your choice for the resource group
3. Region : Region where you want to deploy the resource group

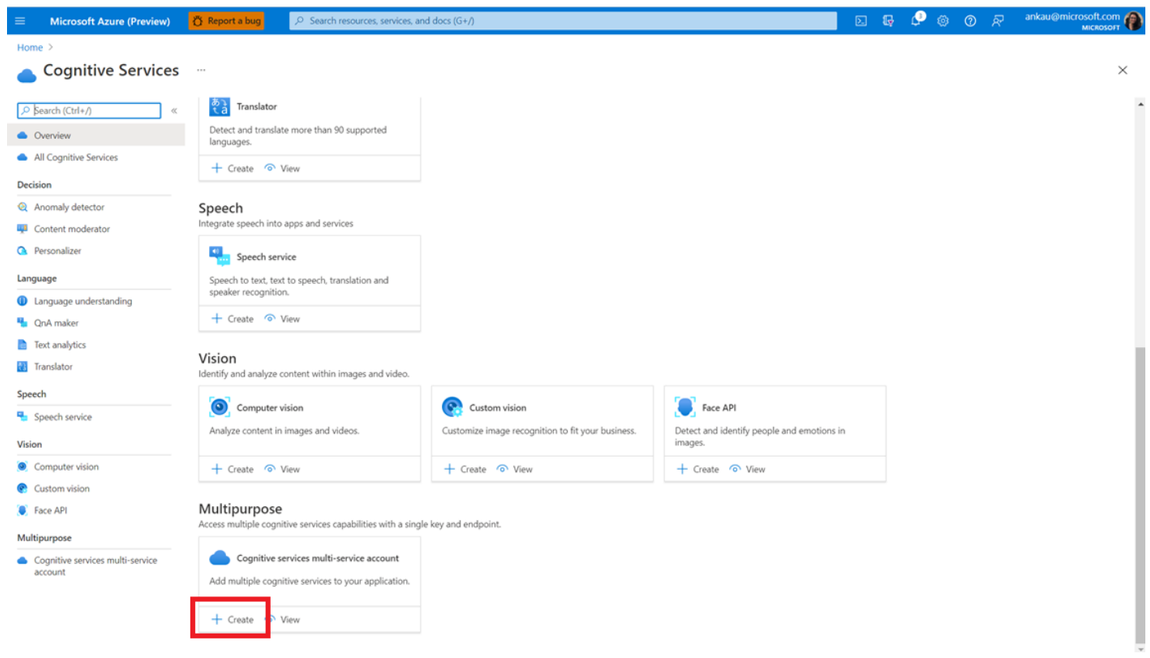
Click Review + Create.



<u>Create Cognitive Service Resource</u>

Once the resource group is created, search for Cognitive Services in the search bar above and select Cognitive Services.

You can skip this step if you already have a Cognitive Service in place for Text Analytics. This can be a multipurpose Azure Cognitive Resource or a Text Analytics Resource.

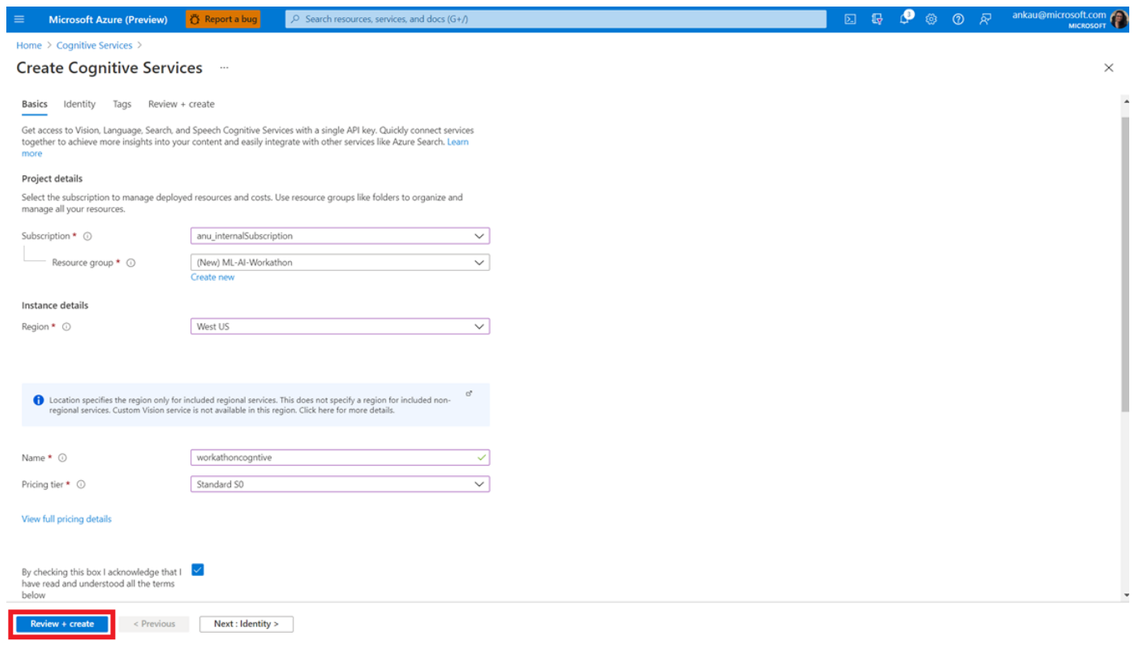


Create a multipurpose cognitive service

Significance : A multipurpose Cognitive Service account allows you to leverage the same resource for many cognitive services, which include :

Computer Vision - Analyze images
Content Moderator - Check text, image or videos for offensive or undesirable content
Face - Recognize people and their attributes in an image
Form Recognizer - Identify and extract text, key/value pairs and table data from form documents
Language Understanding - Extract meaning from natural language
Speech - Transform speech-to-text, text-to-speech and recognize speakers
Text Analytics - Detect sentiment, key phrases, entities and human language type in text

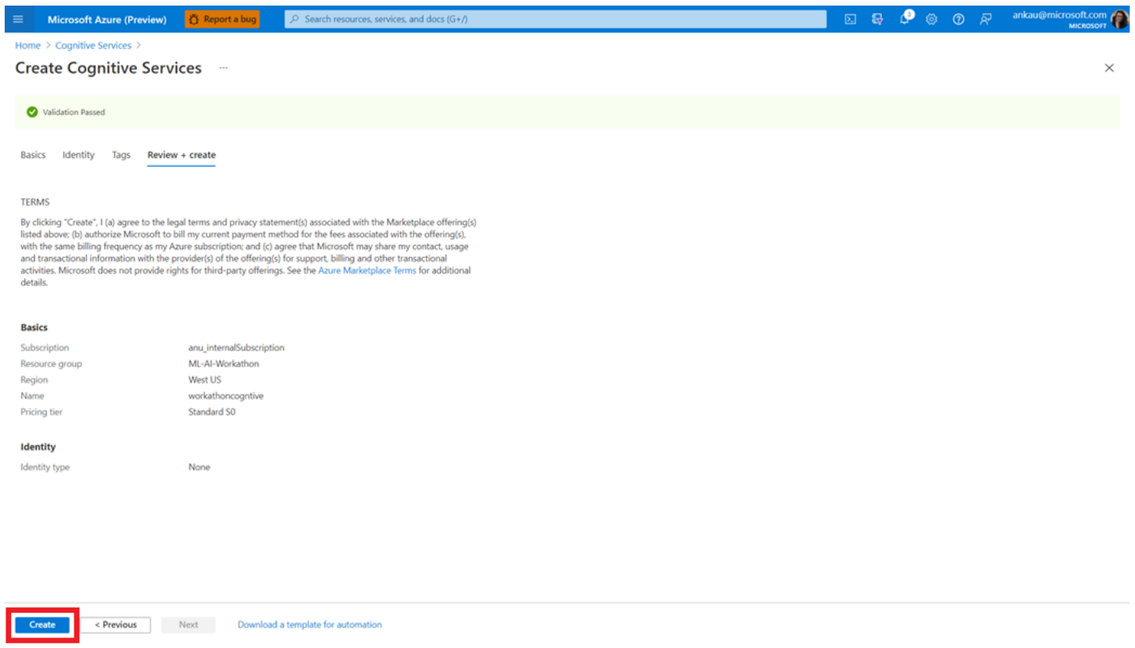
In this lab, we used a multipurpose Cognitive Service account since we would be learning about all the above-mentioned services.
However, you can also spin up individual services to execute these labs or for your development / production scenarios. The only difference is spinning up individual services allows logical separation from workspace standpoint and easy monitoring of billability.



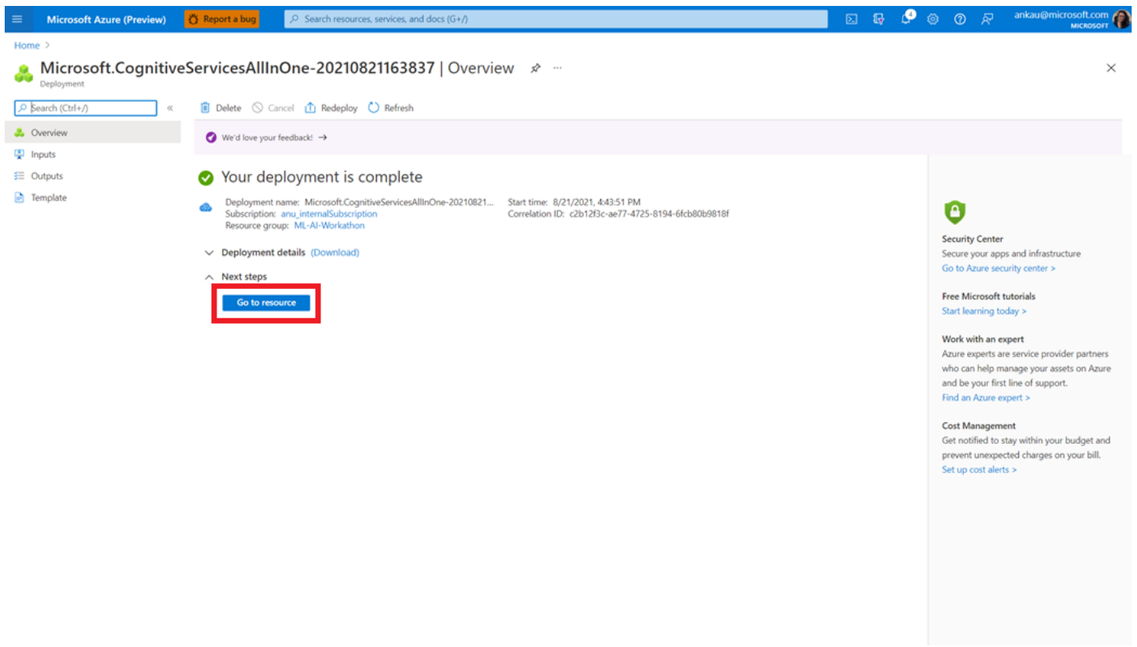
Enter the details to create a new cognitive service as follows -

Project details	Description
Subscription	Select one of your available Azure subscriptions.
Resource group	The Azure resource group that will contain your Cognitive Services resource. You can create a new group or add it to a pre-existing group.
Region	The location of your cognitive service instance. Different locations may introduce latency but have no impact on the runtime availability of your resource.
Name	A descriptive name for your cognitive services resource.
Pricing tier	The cost of your Cognitive Services account depends on the options you choose and your usage.

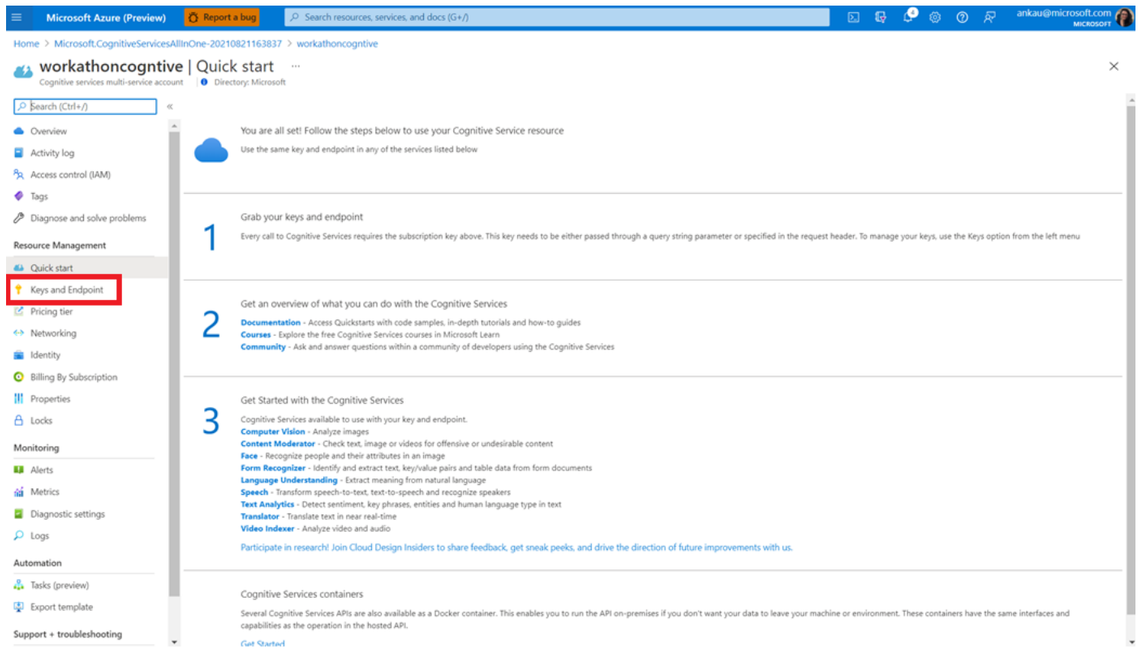
Click Review + Create.



Verify the details and click Create.



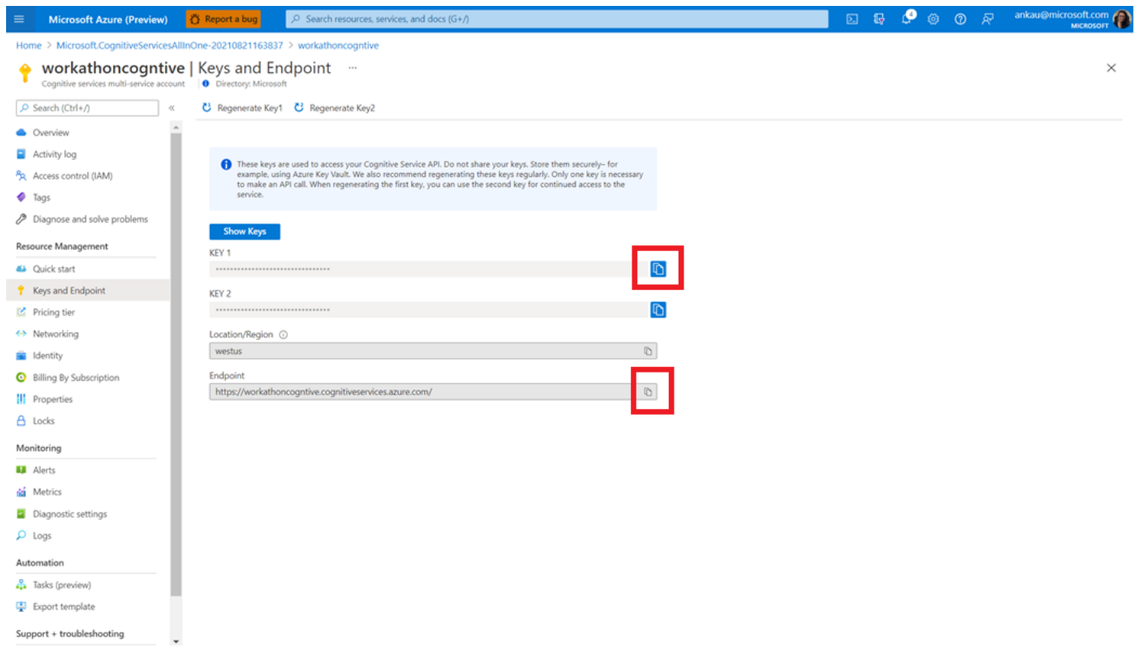
After the resource has been deployed, click Go to Resource.



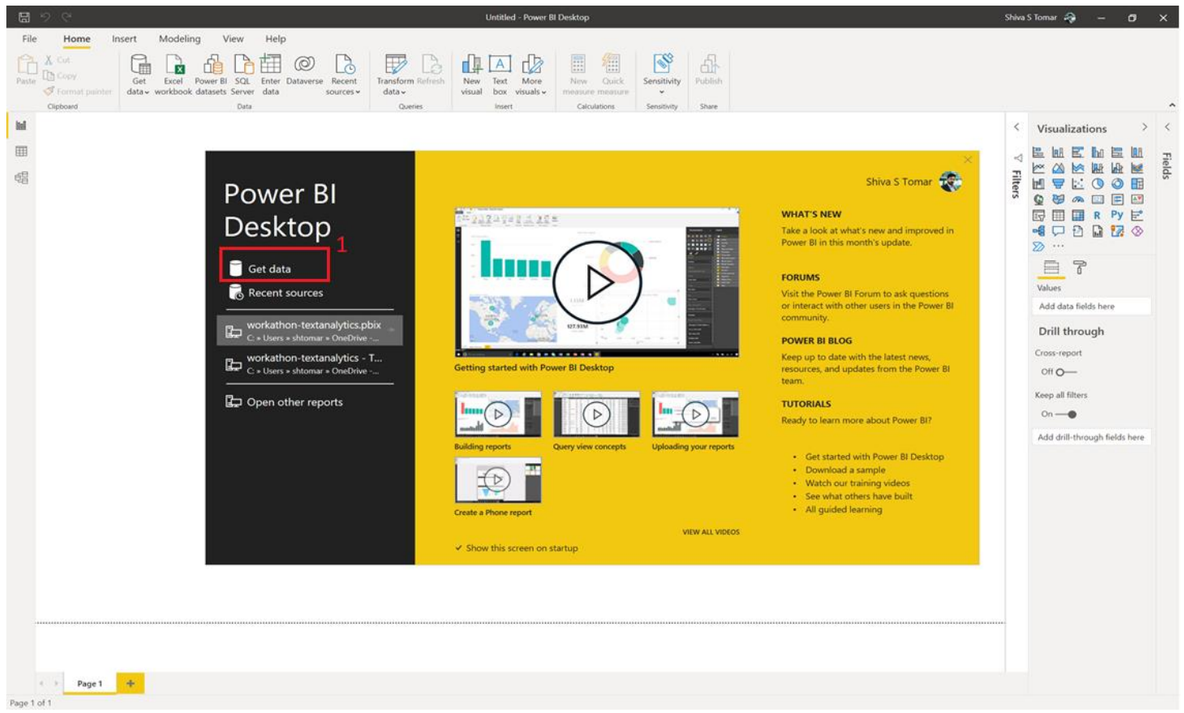
<u>Copy keys & endpoints</u>

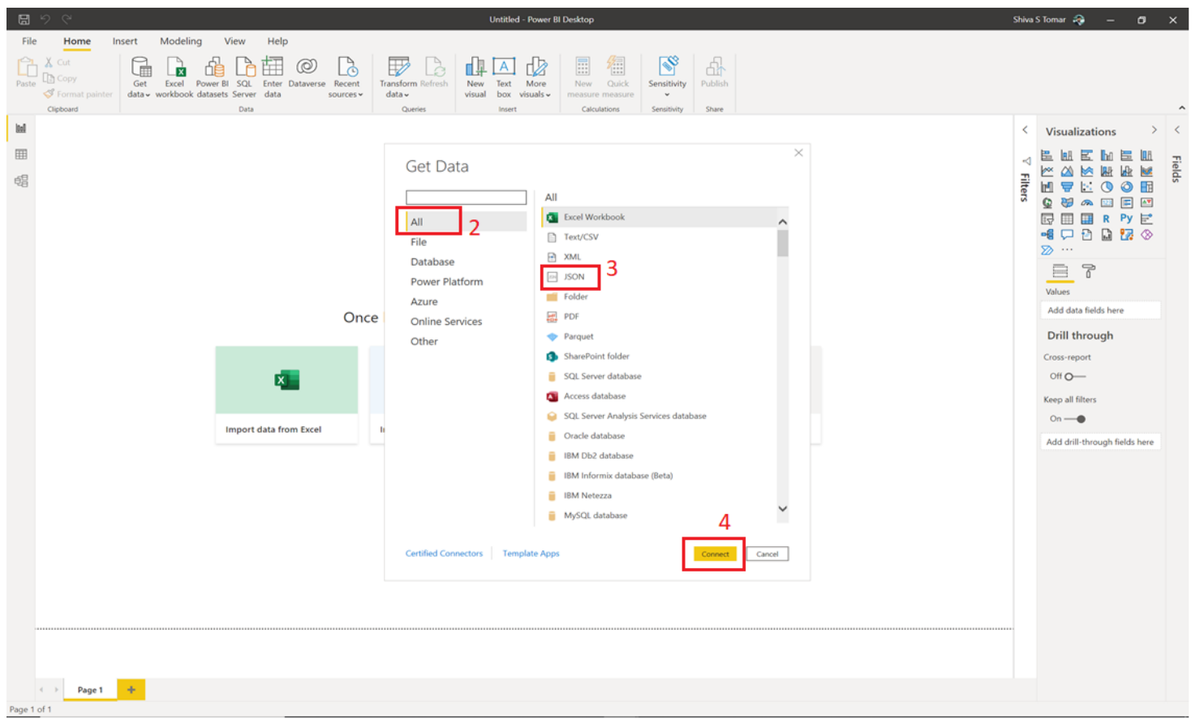
On the Quick start page, you can find details about different cognitive services and can click the hyperlinks to learn more.

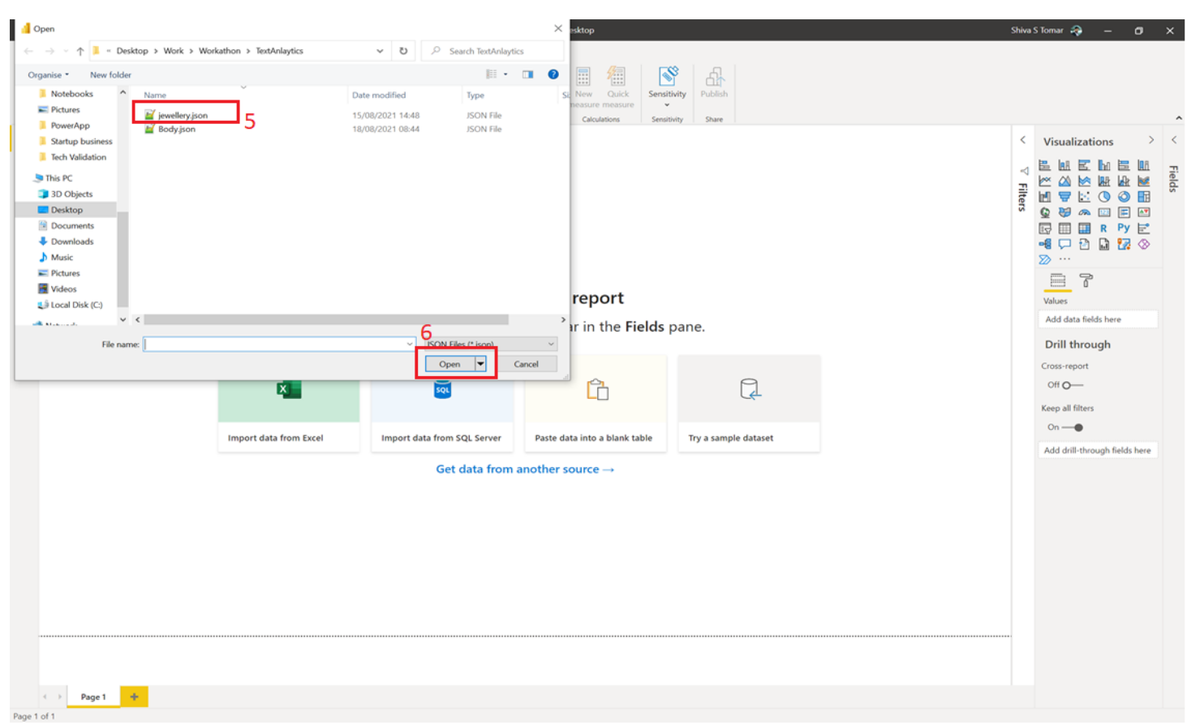
Click Key and Endpoints.



Copy the Key and Endpoint. Paste these in a notepad. You will leverage these in the next steps.







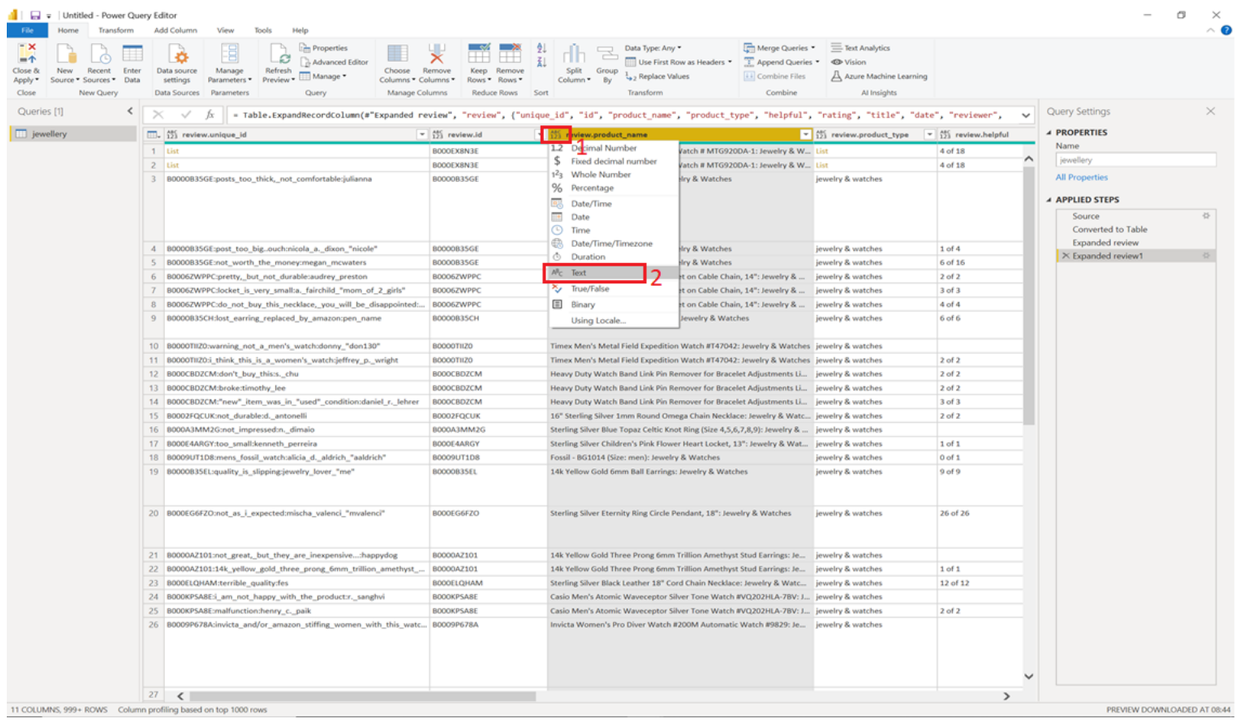
Build Power BI Report

We will now switch to PowerBI Desktop to create the report.
We will start with creating a dataset named Jewellery.

1. Select Get data on the home screen

2. Select All tab
3. Select JSON
4. Click Connect

5. In your computer, browse to the file you downloaded
6. Click Open

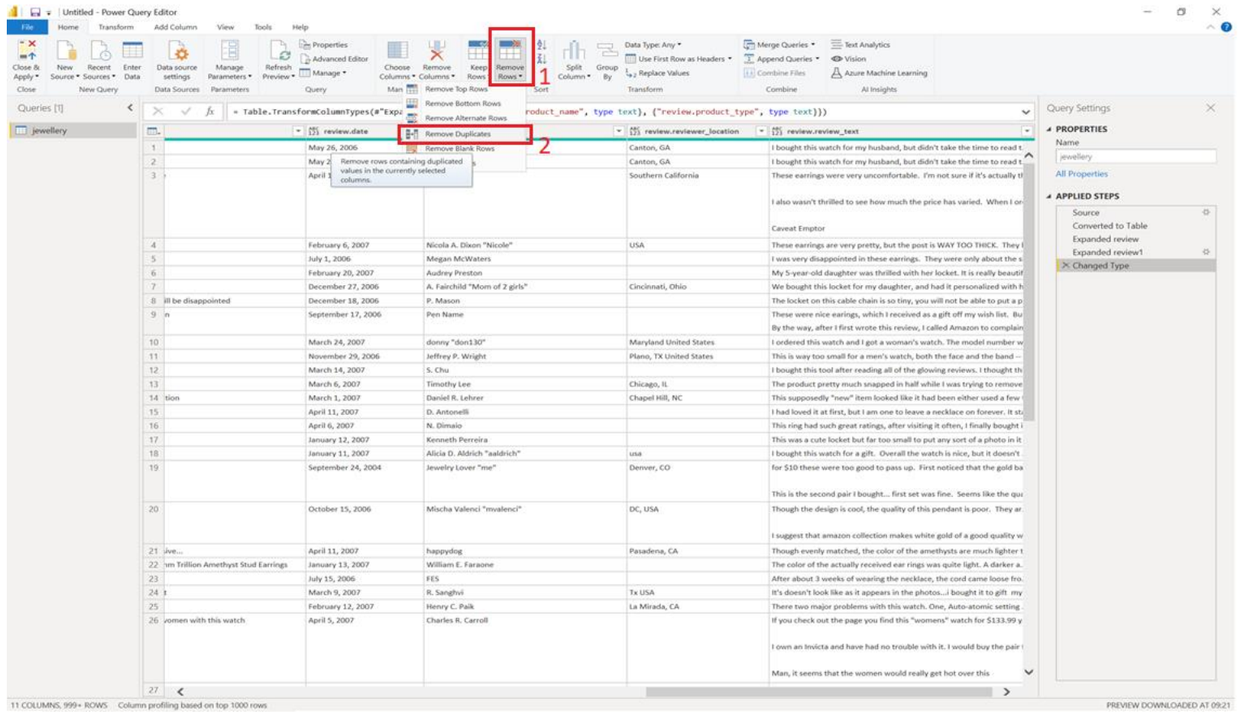


Editing the File in Query Editor

Once you upload the JSON file, it will be opened in the query editor. We will now prepare and clean the data before performing text analytics and translation operations on top of it.

Change Column Type

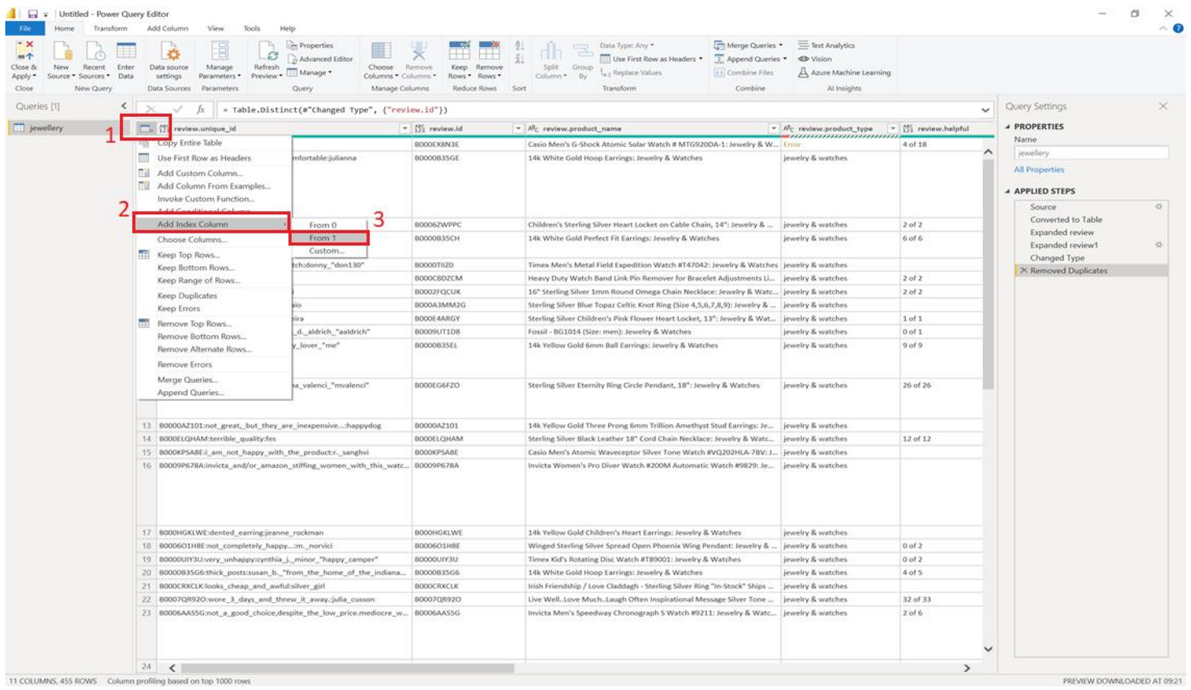
1. Select the Column Type icon for review.product_name
2. Select Text

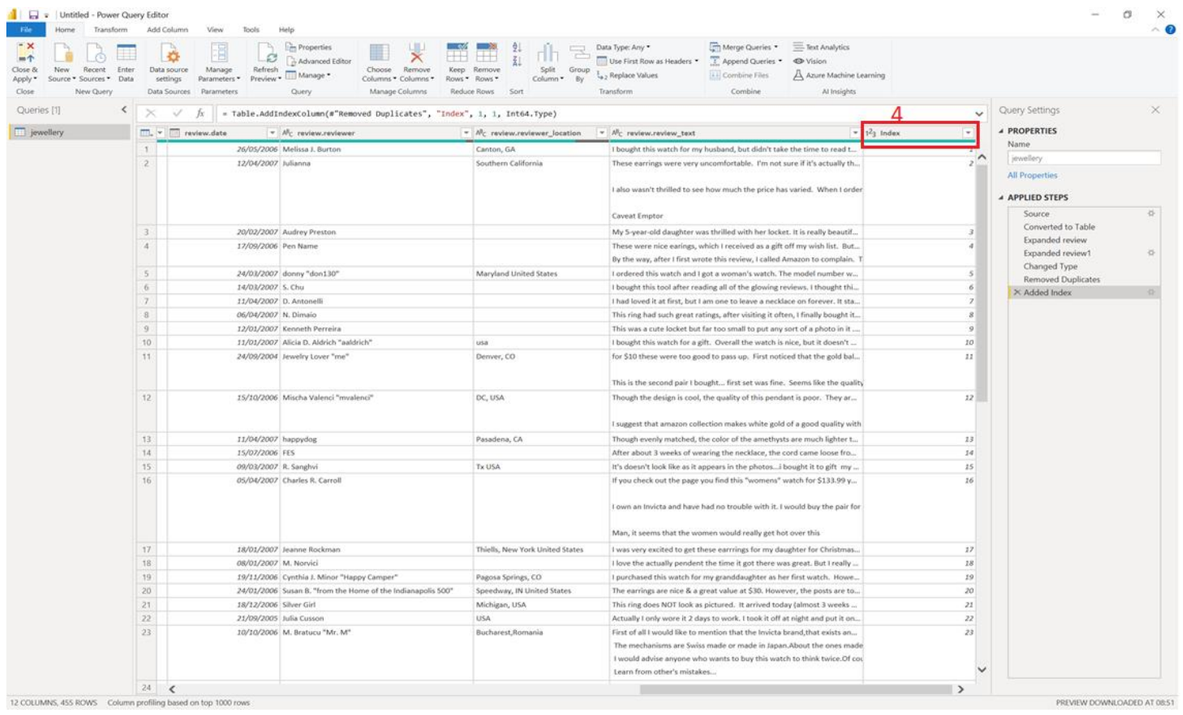


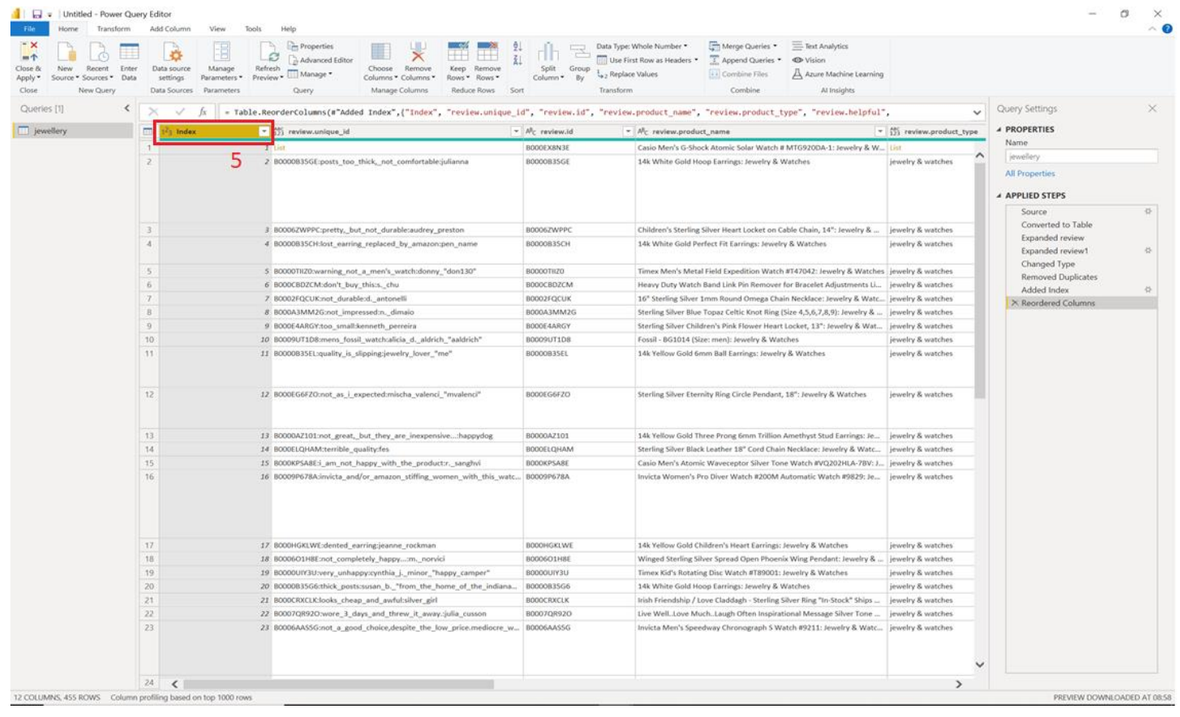
Remove duplicates

We will now remove duplicates basis the review.id column.

1. Select the review.id column
2. Click remove rows > Remove duplicates







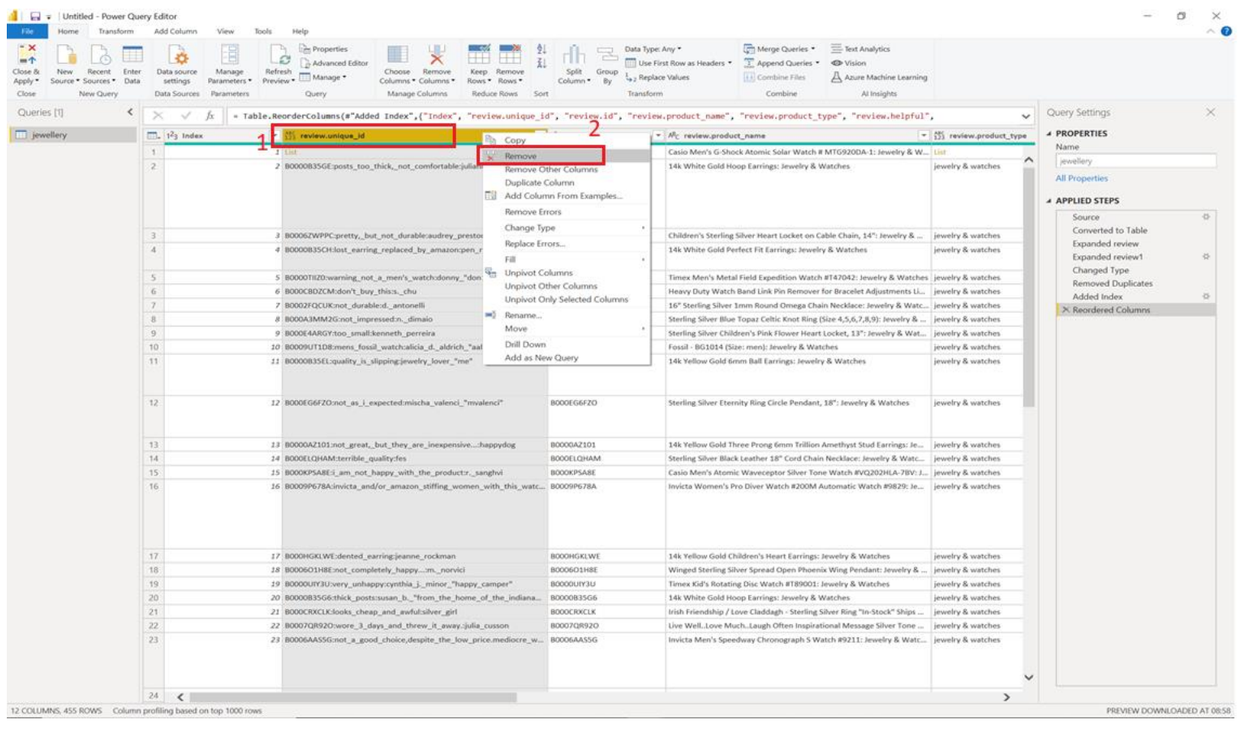
Add Index

We will now add a unique index that will be used to create relationships with other datasets going forward.

1. Select the icon at the top left corner of the table
2. Select Add Index Column
3. Select ‘From 1’ [This will start the indexing from 1]

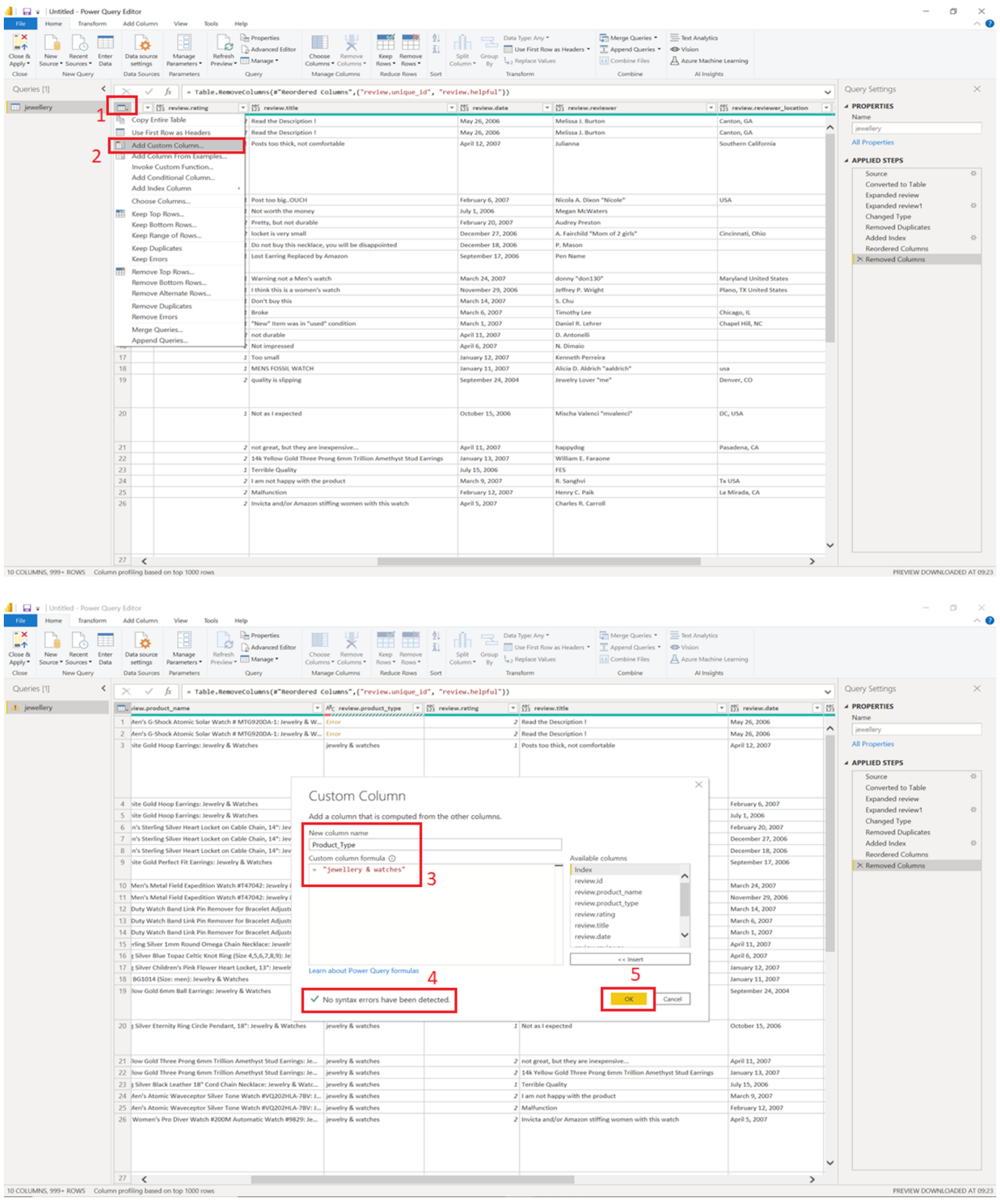
4. Notice the position of Index column in the Dataset. It gets added at the end of the Dataset. Let’s bring it to the front. Right click on Index column > Move > To Beginning OR you can drag it all the way to the beginning.

5. The Index column is now in the beginning of the dataset



Remove columns

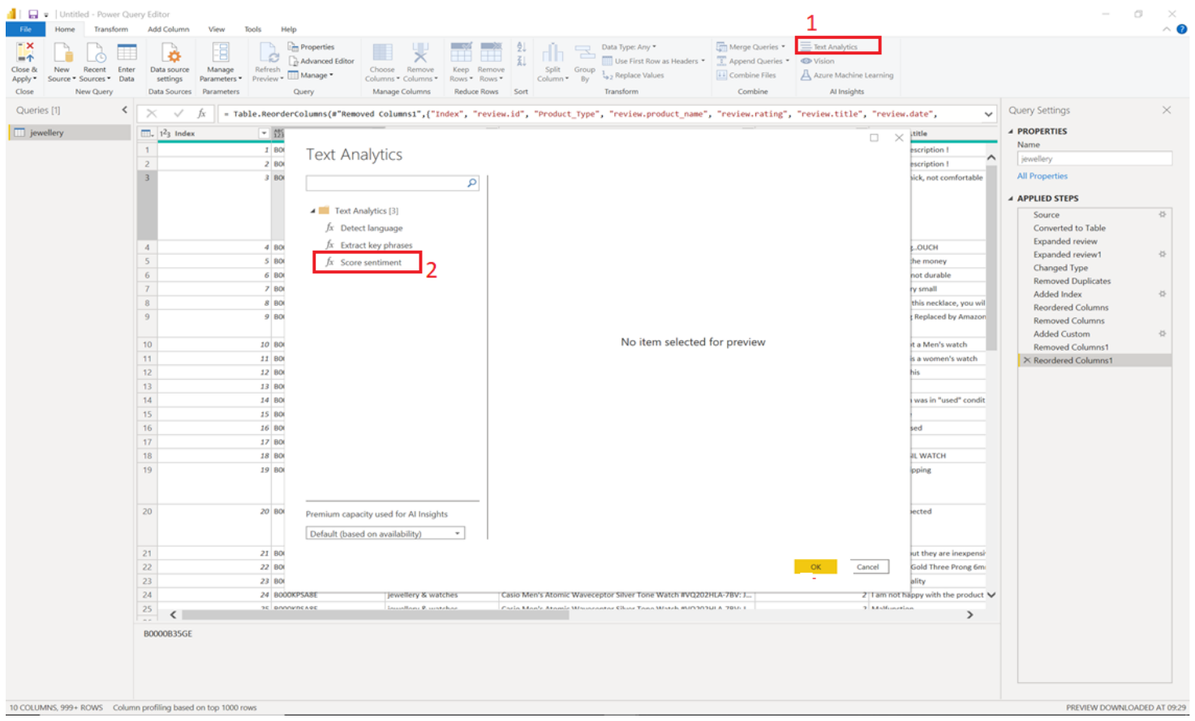
1. Right click on the review.unique_id column
2. Select Remove

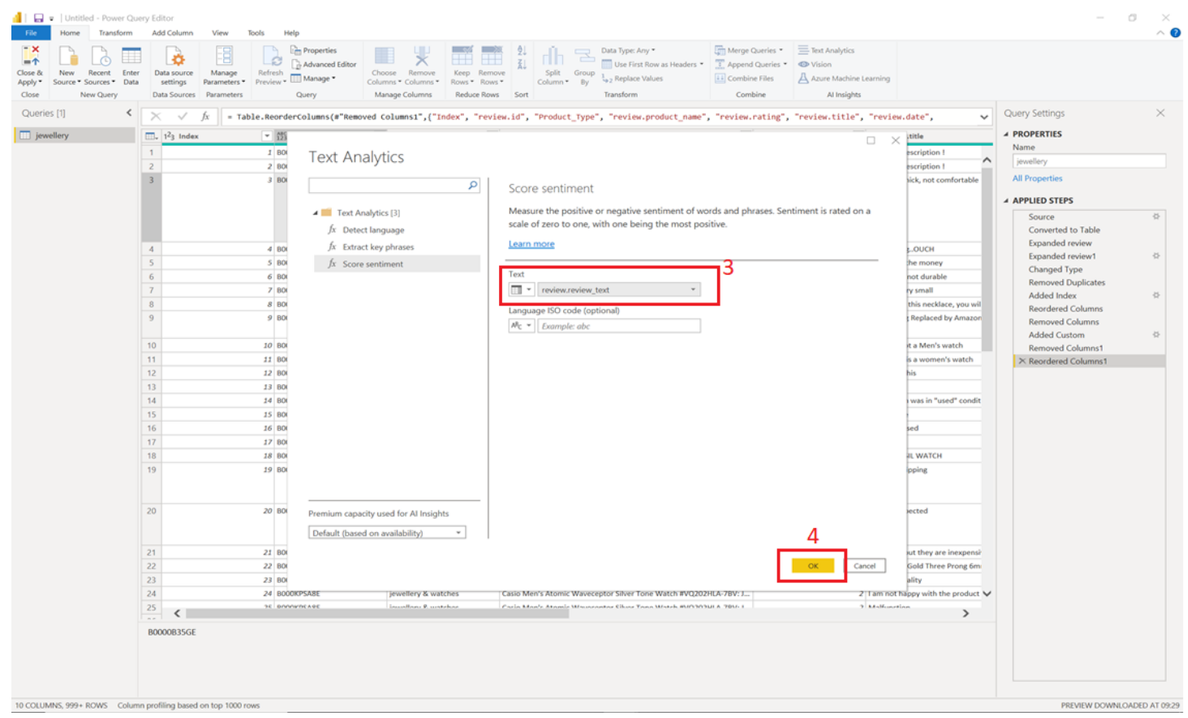


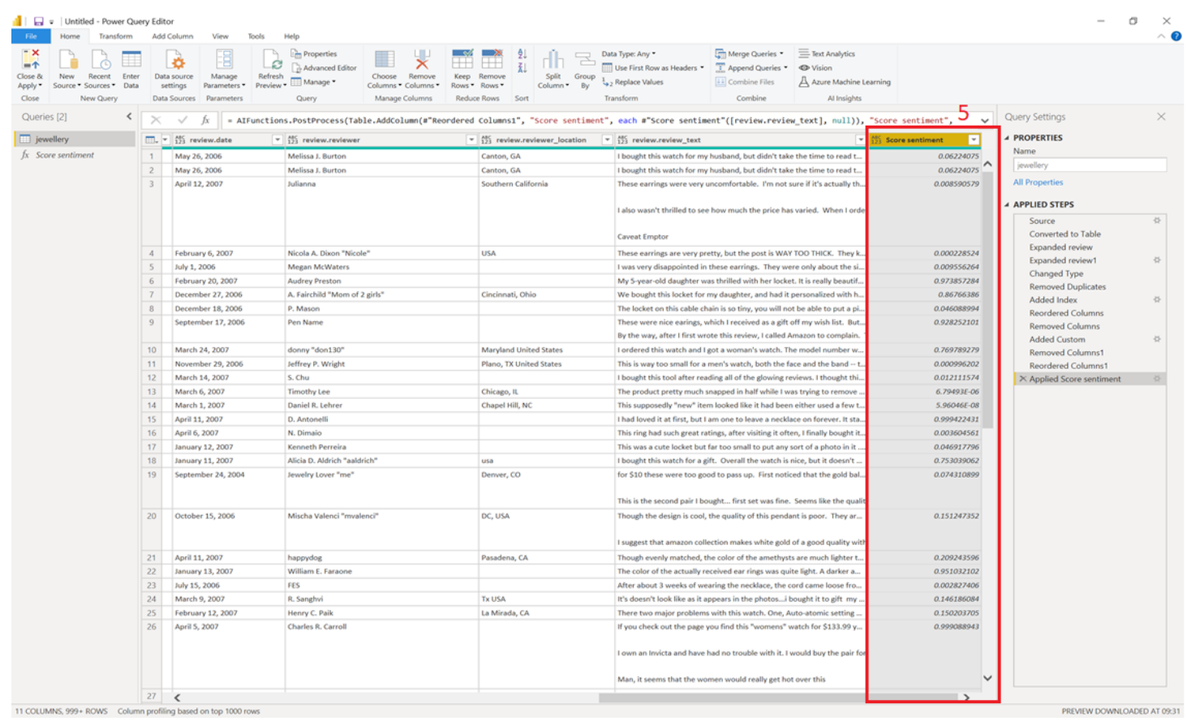
Add a new custom column

1. Click on the icon at the top left corner of the dataset
2. Select Add Custom Column

3. New column name : Product Type
   Custom column formula : "Jewellery & Watches"
4. Make sure there are no syntax errors
5. Click OK







Derive Sentiment Score

We will now use the built-in Text Analytics capabilities to extract the Sentiment score of the customer reviews.

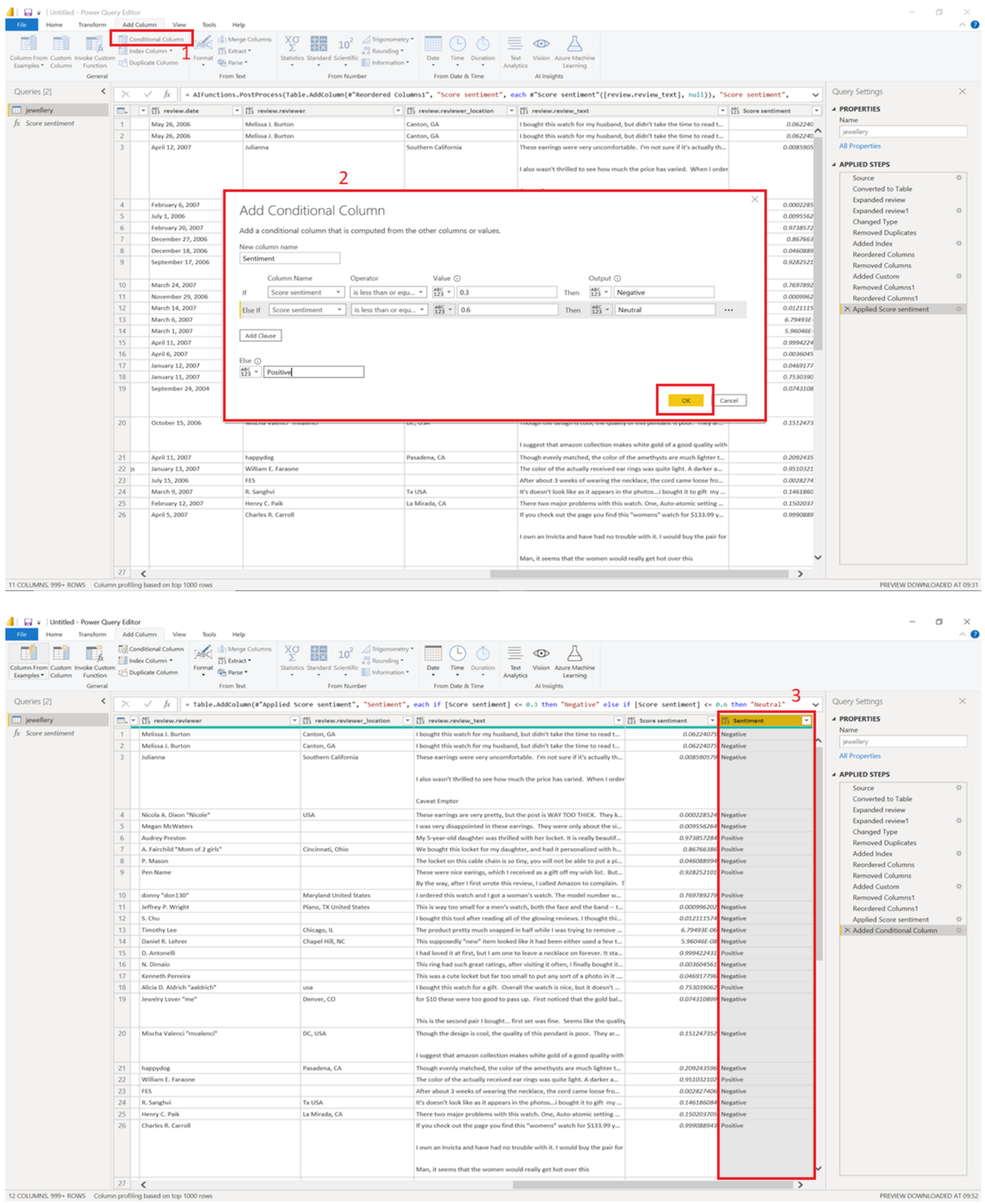
The current integration of Text Analytics in PowerBI, provide only 2 functionalities :

1. Extract Key Phrases
2. Score sentiment

We are leveraging the 'Score sentiment' capability from here. To perform other operation (Entity Extraction, PII extraction, summarization etc) you will have to write custom function and make API calls using it. We will be writing the advanced custom queries for entity extraction and Translation going forward in the workshop.

1. Select Text Analytics in the query editor
2. Select Sentiment score
3. Text : review.review_text
4. Click Ok
5. This will take a while to execute and you will see the Score Sentiment column added to the dataset

Note : If asked to authenticate, enter your cognitive service endpoint and key as copied earlier.



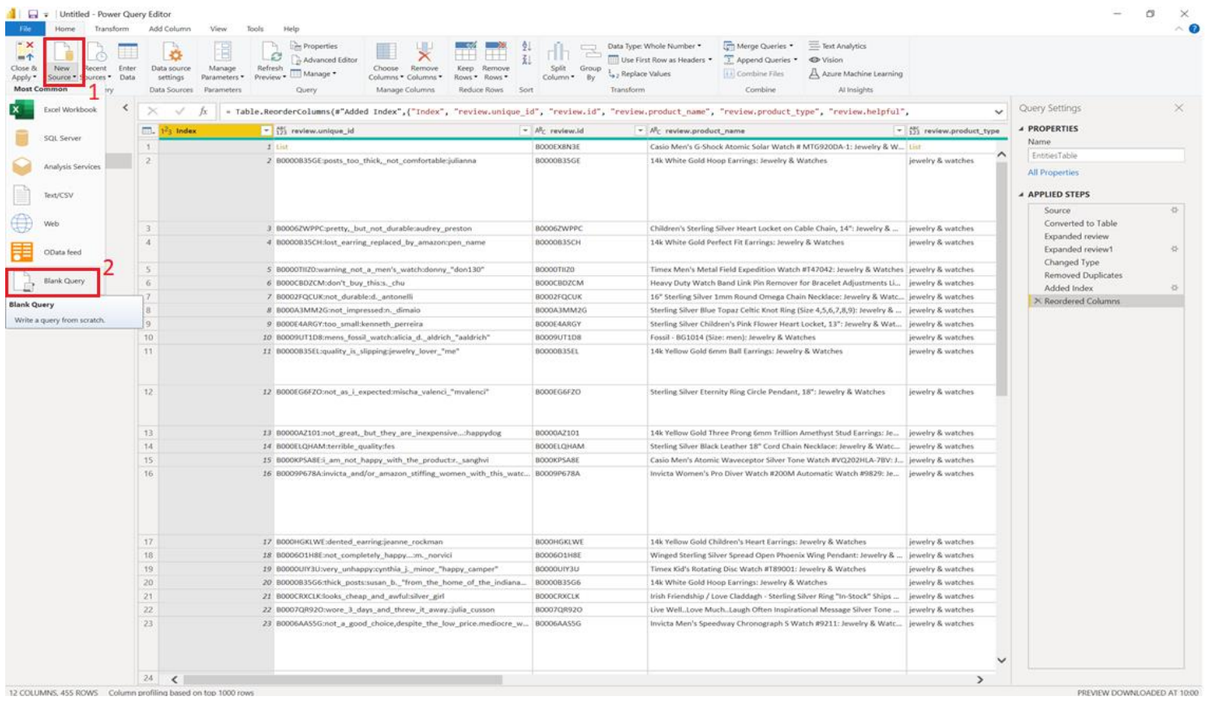
Add conditional Column

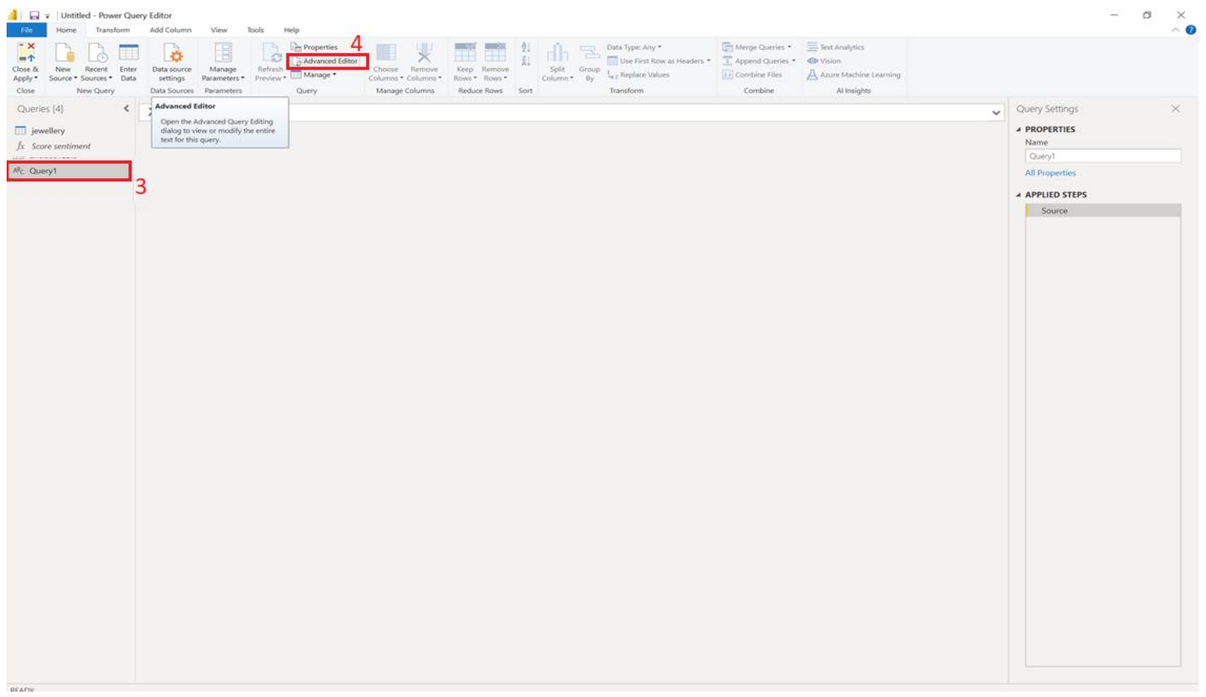
We will now create a column for the sentiment basis the sentiment score we derived.

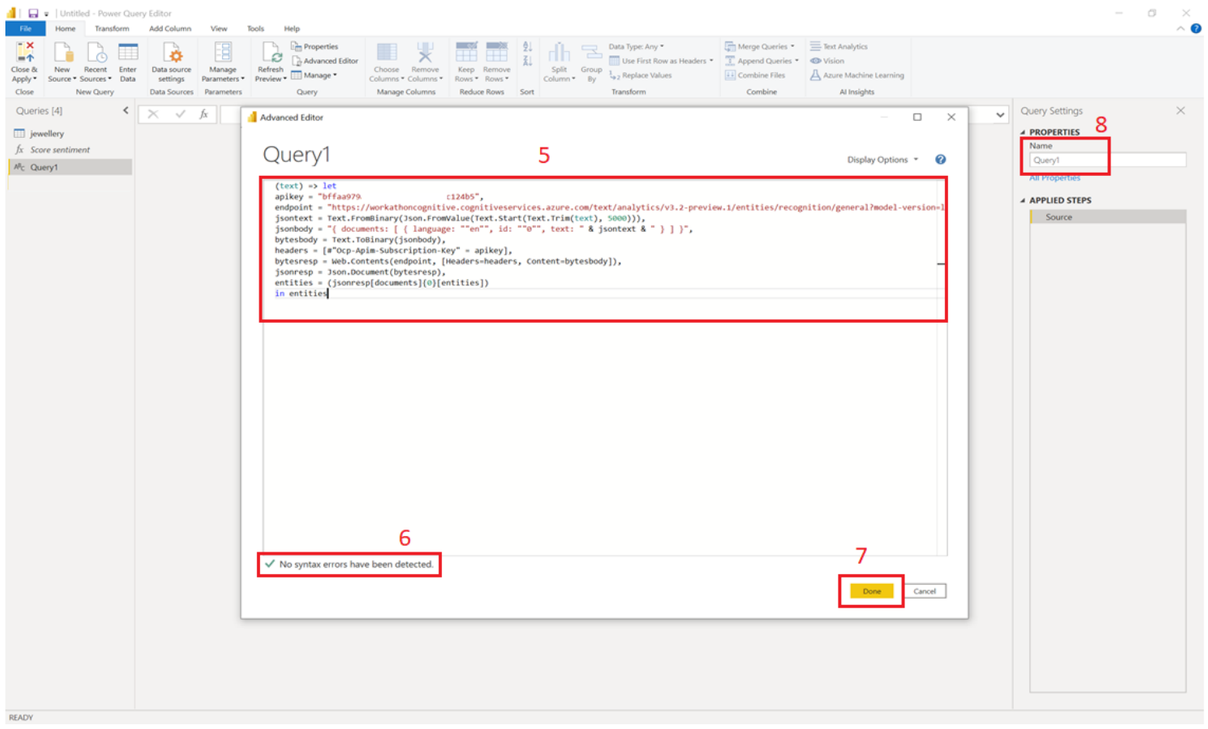
1. Select Conditional Column in Add Column tab

2. Fill in the fields basis the following inputs and also refer the screenshot :
   if 'Score sentiment' column <= 0.3 then output = Negative
   Else if 'Score sentiment' column <= 0.6 then output = Neutral
   Else Positive

3. Once added, you will see the Sentiment column as highlighted.







Create custom query to extract entities

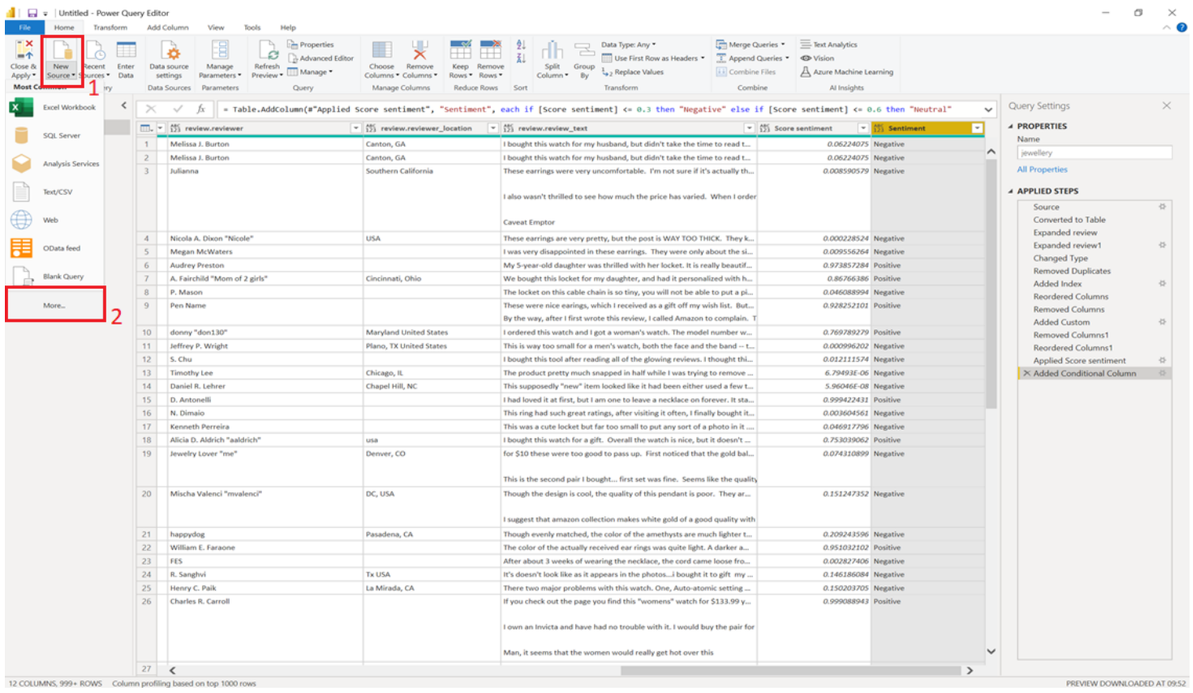
As mentioned above in the Sentiment section, since entity extraction is not available as an out of the box functionality in Power BI, we will be writing a custom query to invoke the Text Analytics Entity Extraction API.

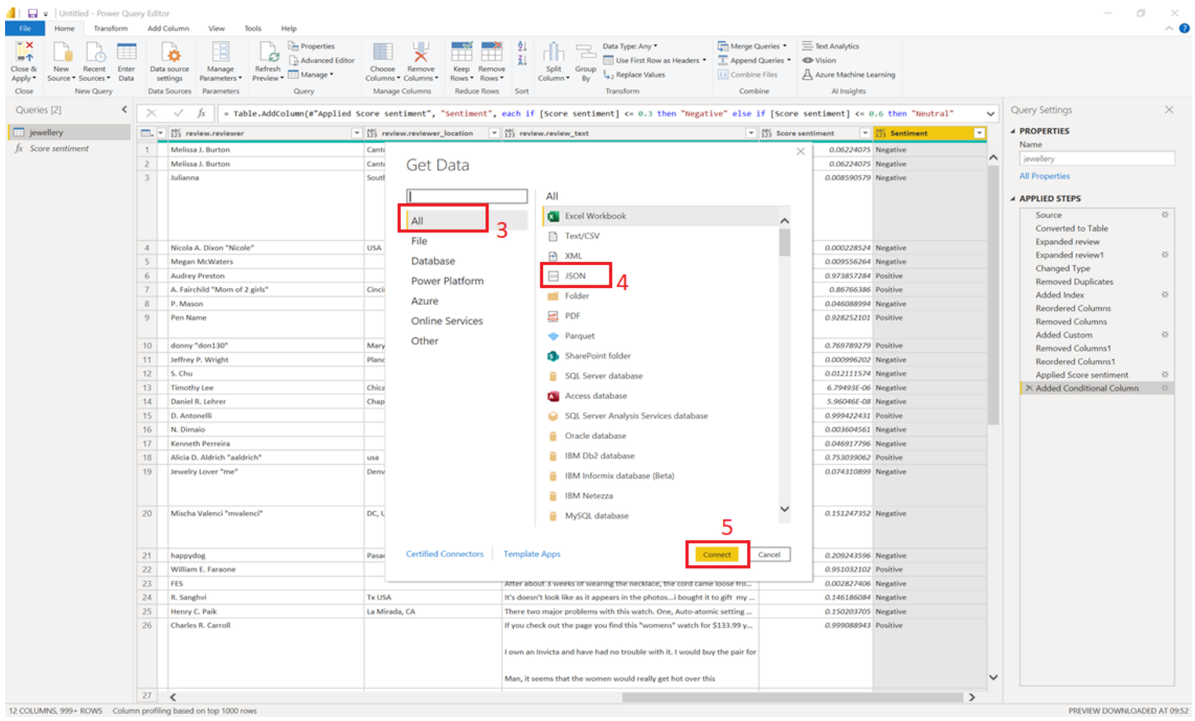
1. In query editor, select New Source
2. Select blank query

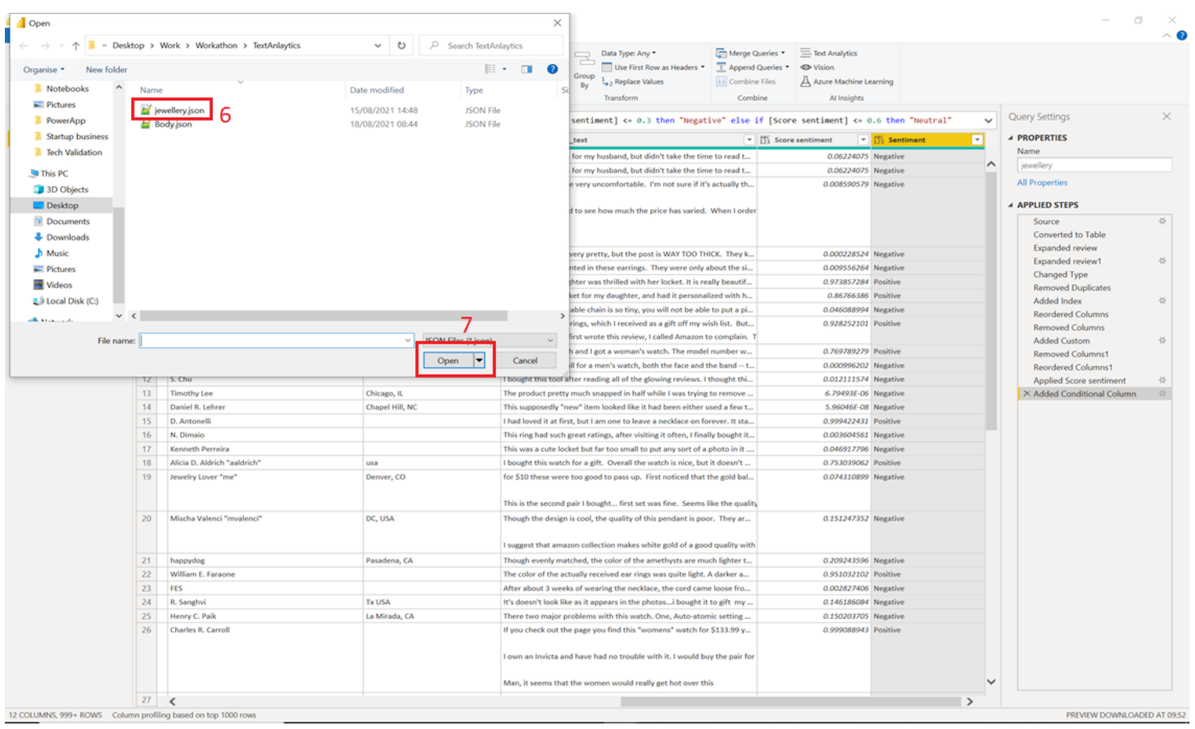
3. Select Query 1
4. Select Advanced Editor
5. Paste the following query in the editor. Make sure to add your API key within quotes for apikey value and replace underlined portion in endpoint with your cognitive service endpoint.

```
(text) => let
apikey = "<<paste your API Key here>>",
endpoint = "https://workathoncognitive.cognitiveservices.azure.com/text/analytics/v3.2-preview.1/entities/recognition/general?model-version=latest&showStats=false&loggingOptOut=false&stringIndexType=TextElement_v8",
jsontext = Text.FromBinary(Json.FromValue(Text.Start(Text.Trim(text), 5000))),
jsonbody = "{ documents: [ { language: ""en"", id: ""0"", text: " & jsontext & " } ] }",
bytesbody = Text.ToBinary(jsonbody),
headers = [#"Ocp-Apim-Subscription-Key" = apikey],
bytesresp = Web.Contents(endpoint, [Headers=headers, Content=bytesbody]),
jsonresp = Json.Document(bytesresp),
entities = (jsonresp[documents]{0}[entities])
in entities
```

6. Make sure there are no syntax errors
7. Click Done
8. Rename Query1 to entities







Add a new source for the Entities

Notice in this section, we are reading from the same JSON source again. The reason to do so is that - In PowerBI, when you try to run the built-in function and custom function on the same dataset, you might get the following error :

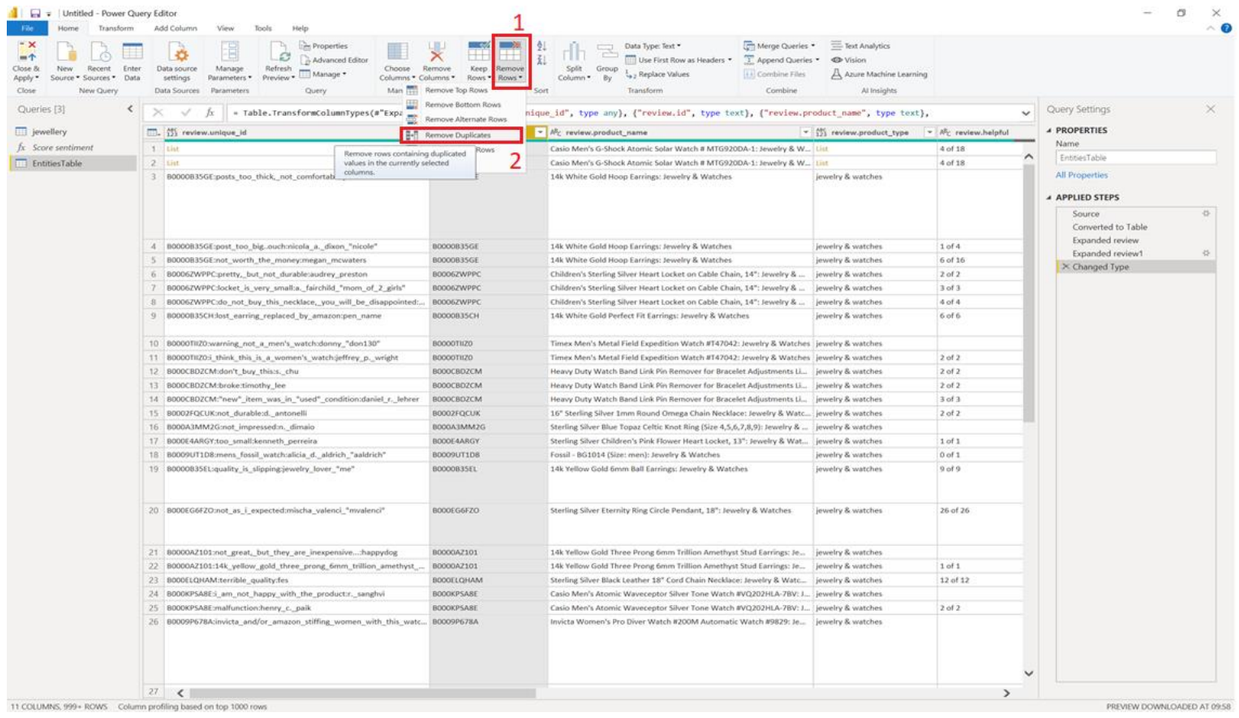
> ⚠ Formula.Firewall: Query 'jewellery' (step 'Invoked Custom Function') references other queries or steps, so it may not directly access a data source. Please rebuild this data combination.

We will be leveraging this dataset to make build the custom functions for the following :

a. 'Entities' – to extract entities using Text Analytics from the review text [custom query created in the above step]
b. 'Translated Output' – to translate the incoming Reviews & Entities from English to Hindi [we will create this function in the next steps]

Follow the steps as highlighted to create a new dataset :

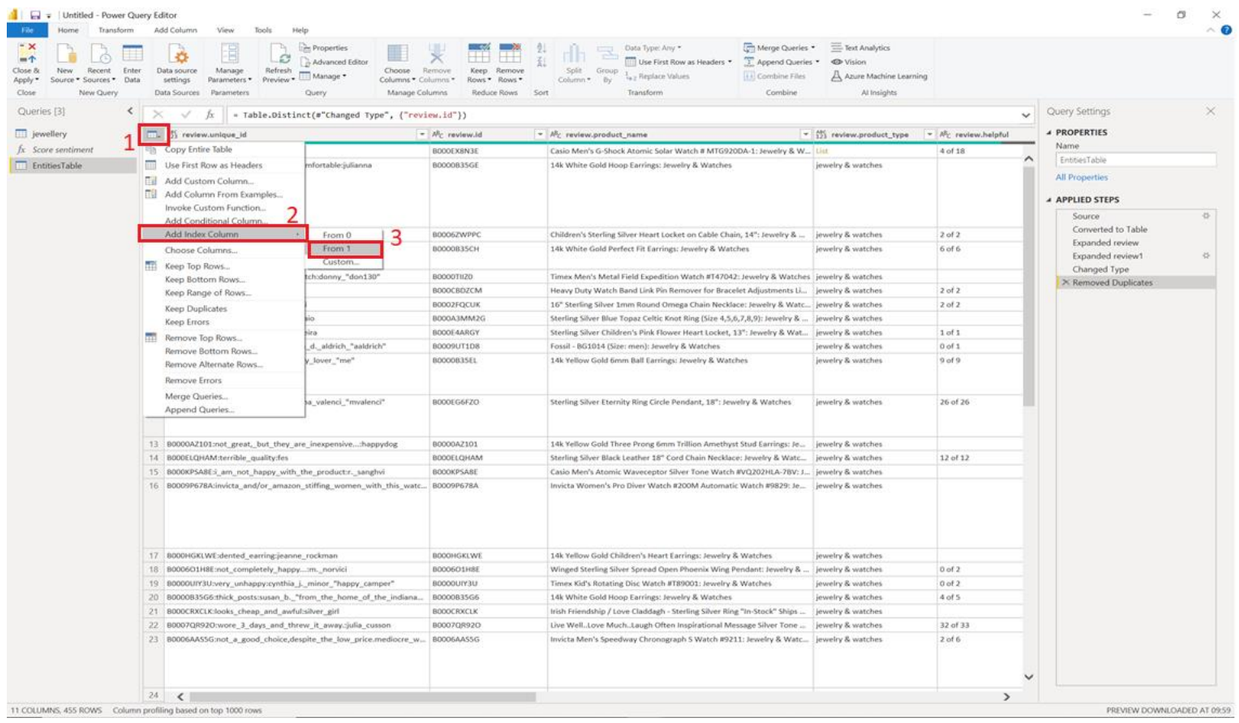
1. In the query editor, select New Source
2. Click More
3. Select All tab
4. Select JSON
5. Click Connect
6. In your computer, browse to the file you want to upload
7. Click Open
8. Rename this dataset to 'Entities'



Remove duplicates

We will now remove duplicates basis the review.id column.

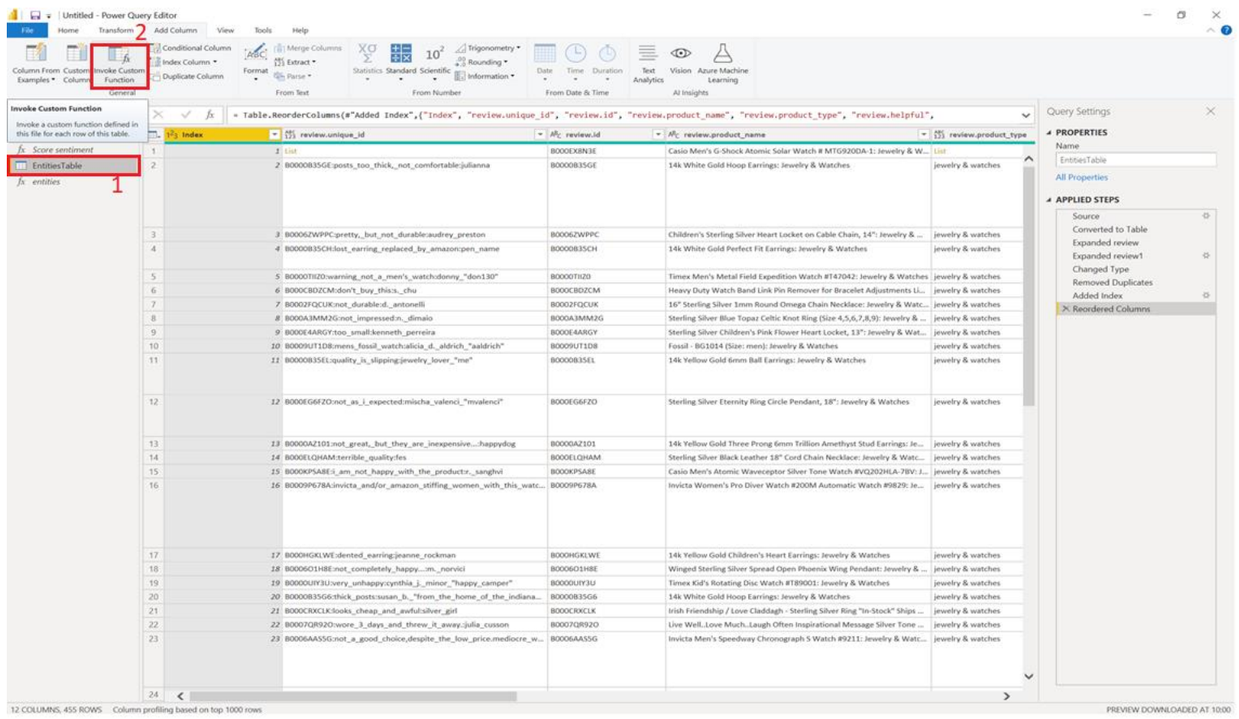
1. Select the review.id column
2. Click remove rows > Remove duplicates

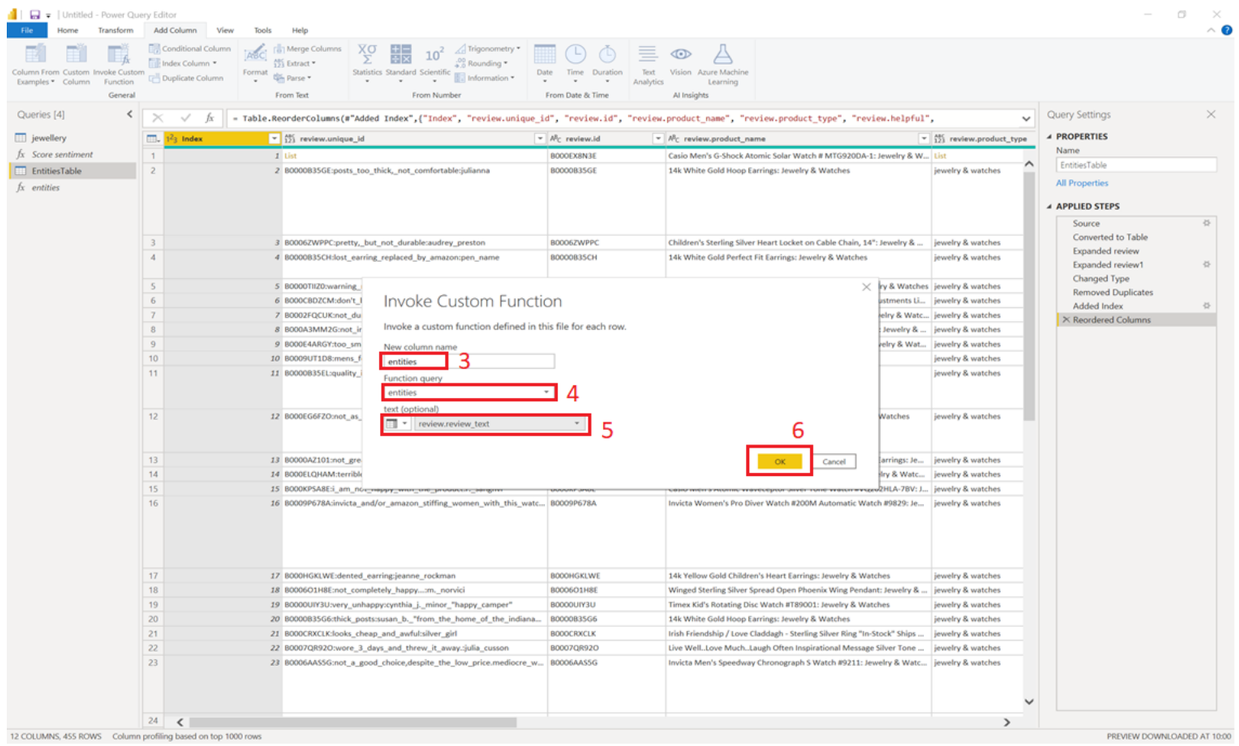


Add Index

We will now add a unique index like we created for the previous dataset, that will be used to create relationships with jewellery dataset going forward.

1. Select the icon at the top left corner of the table
2. Select Add Index Column
3. Select From 1
4. Bring the index column to the front



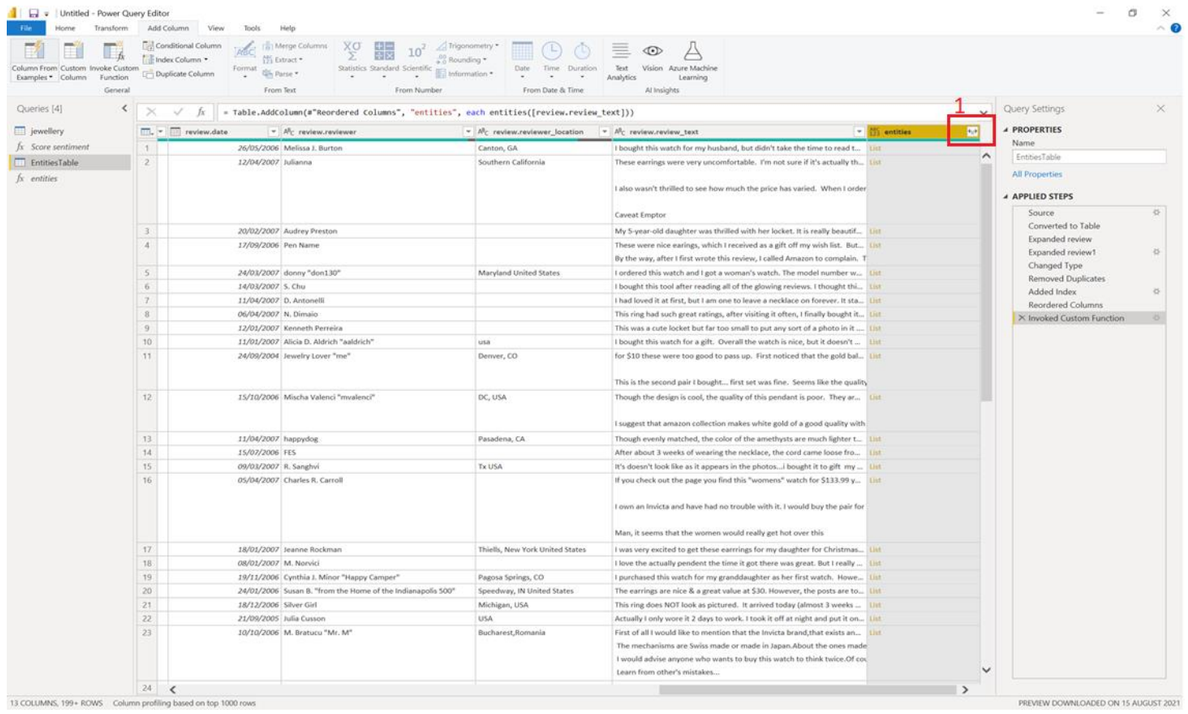


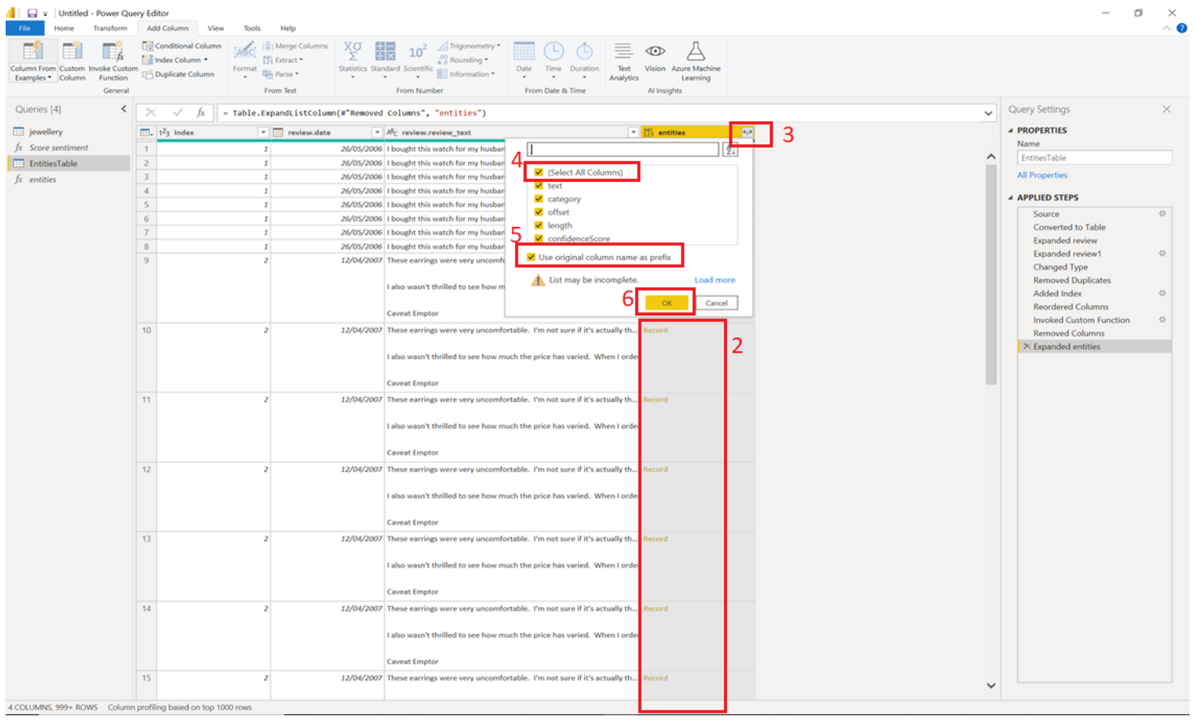
Invoke Custom function to extract entities

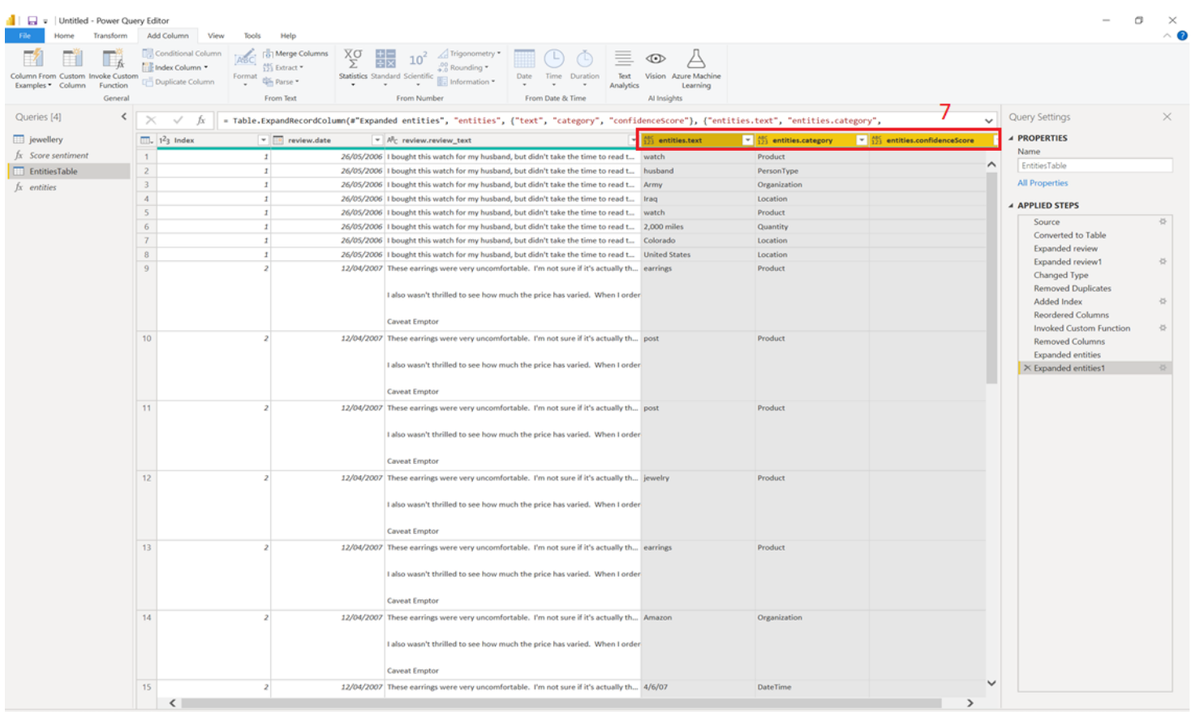
Earlier, we just created the custom function to extract entities, we will now be invoking the function to extract the entities from 'review.review_text' column in a new column named 'entities'.

1. Select the Entities table
2. Select Invoke Custom Function
3. New column name : entities
4. Function query : entities [From the dropdown select the name of the function you created above]
5. Text : review.review_text [This is the column that contains the customer reviews text]
6. Click OK
7. Wait while the function is being executed.

Note : If asked to authenticate, set the options to anonymous / public, since we have already provided the API key in the query we wrote.







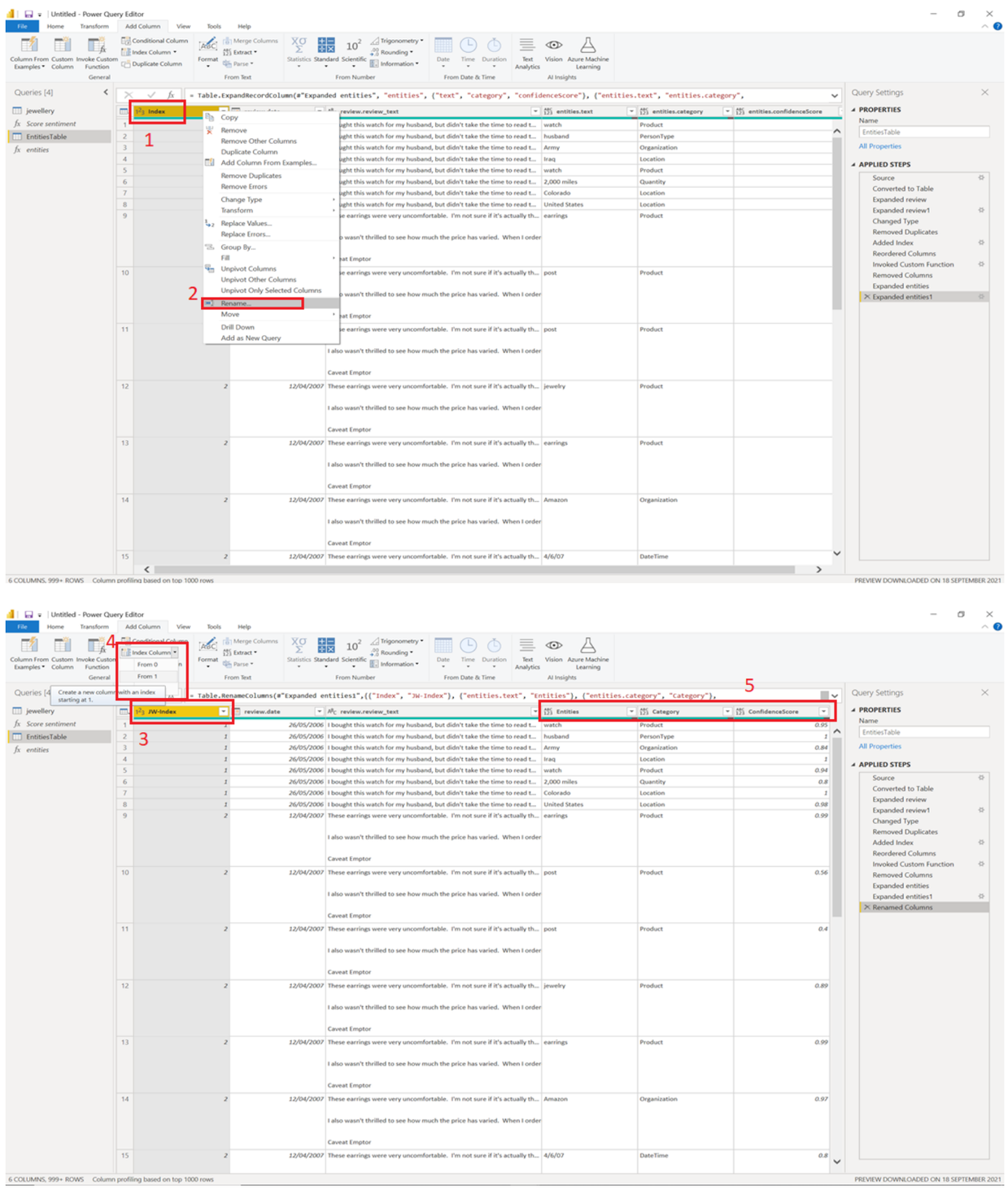
Process entities column

Executing the 'Entities' function will render the results in 'entities' column as 'List' data type. We will need to do some post processing to bring the entities column in a usable format

1. Click the expand button next to entities column.

2. Observe that this changes the column data type from List to Record
3. Again, click the expand button next to entities column
4. Select the checkbox for 'Select all column'
5. Select the checkbox for 'Use original column name as prefix'
6. Click Ok

7. Observe the new columns that are added towards the end of the dataset



<u>Renaming original index and creating new index</u>

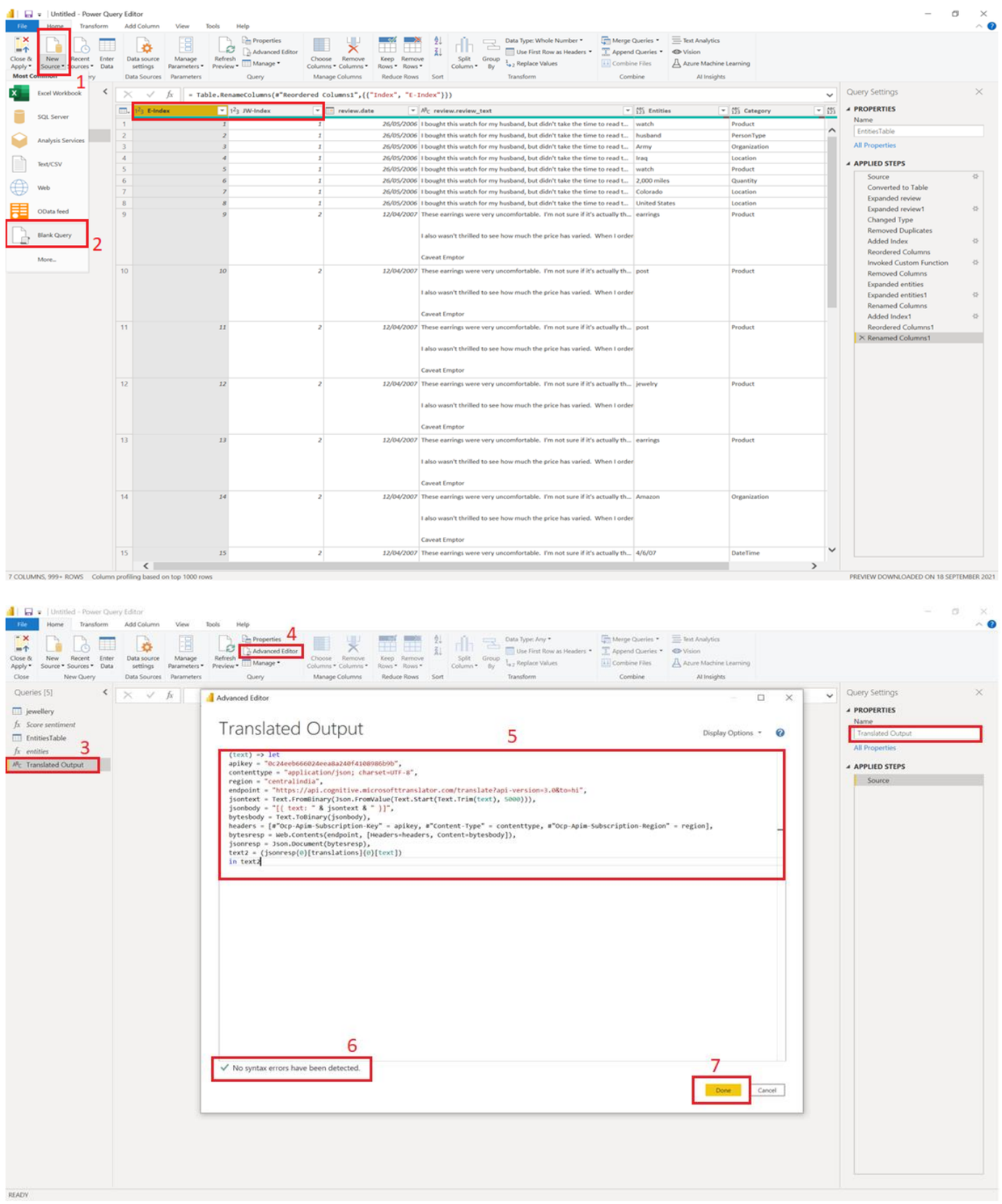
Observe that since multiple entities were extracted from each review text, this lead to duplication of rows, with all the columns being the same except for the columns derived from using custom entity function. Due to this, we do not currently have a primary key for the Entities dataset.

Thus, we will rename the previously created index to JW-Index [since this now acts as a foreign key in Entities Dataset referencing to the primary key (Index) of Jewellery dataset. We can join the 2 tables basis the following condition – Jewellery.index = Entities.JW-Index (1:many)].

We will then add a new index that will be the primary key Entities dataset.

1. Right click on the index column
2. Select Rename

3. Rename to JW-Index
4. Select Index column in Add Index menu and select from 1.
5. Also rename the following columns
   entities.text -> Entities
   entities.category - > Category
   entities.confidencescore -> ConfidenceScore



Create custom query to translate text to Hindi

We will be translating the Customer Reviews and Entities extracted to Hindi, to cater to the audience who want to view the reports in Regional Languages.

You can choose amongst any of the languages supported by Azure Text Translation API. We will be translating the text to Hindi.

Since Text Translation is not available as an out of the box functionality in Power BI, we will be writing a custom function to invoke the Text Translation Azure API.

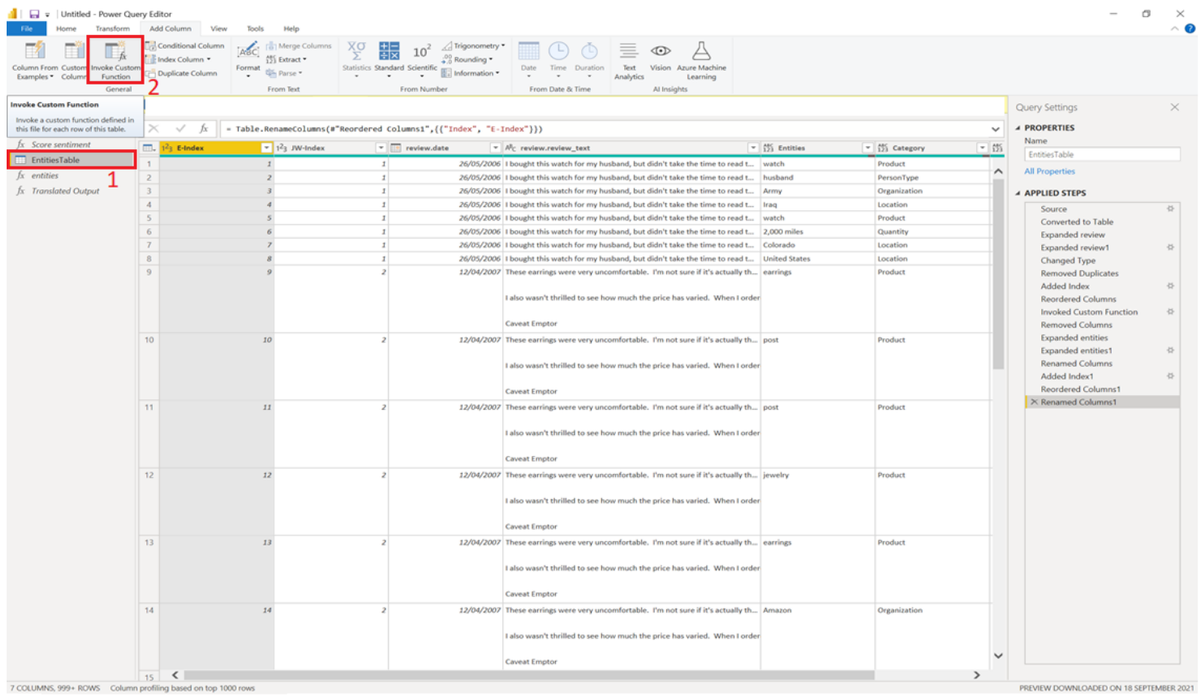
1. In query editor, select New Source
2. Select blank query
3. Select the query and rename to Translated Output
4. Select Advanced Editor
5. Paste the following code in the editor.

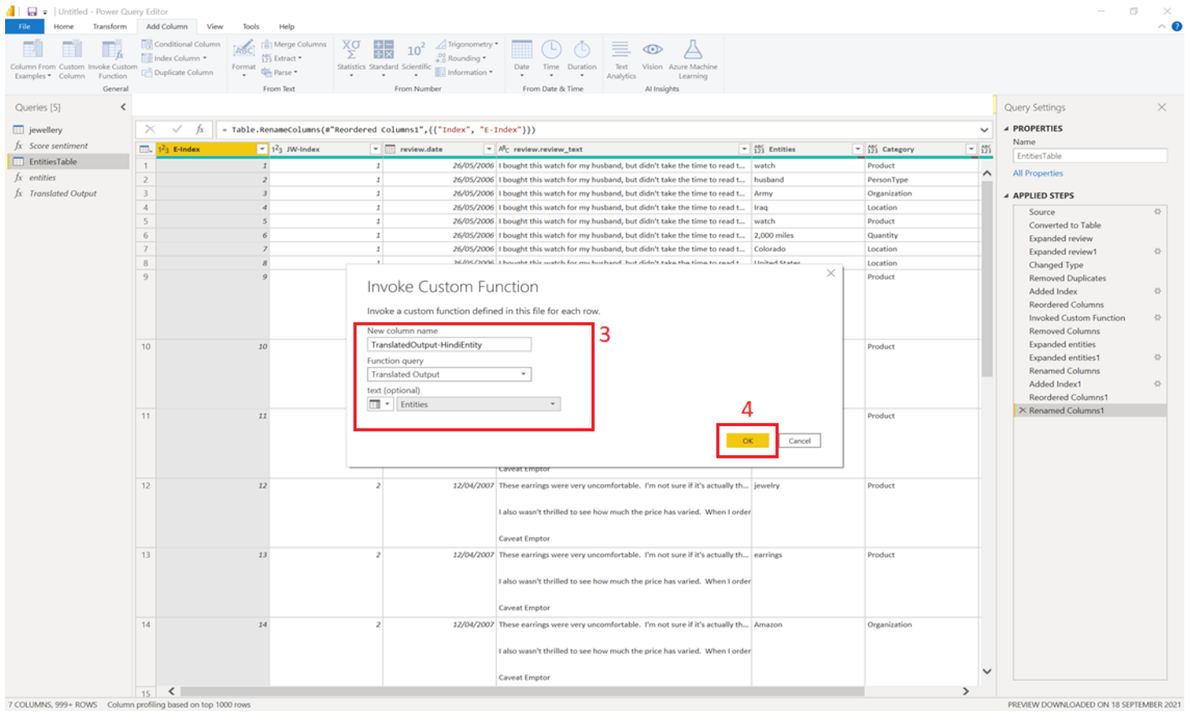
```
(text) => let
apikey = "<<Paste Text Translation API key>>",
contenttype = "application/json; charset=UTF-8",
region = "centralindia",
endpoint = " https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&to=hi&includeSentenceLength=true&Category=da489884-ac06-4fd1-8362-61a836d0791a-SHOPPNG"
jsontext = Text.FromBinary(Json.FromValue(Text.Start(Text.Trim(text), 5000))),
jsonbody = "[{ text: " & jsontext & " }]",
bytesbody = Text.ToBinary(jsonbody),
headers = [#"Ocp-Apim-Subscription-Key" = apikey, #"Content-Type" = contenttype, #"Ocp-Apim-Subscription-Region" = region],
bytesresp = Web.Contents(endpoint, [Headers=headers, Content=bytesbody]),
jsonresp = Json.Document(bytesresp),
text2 = (jsonresp{0}[translations]{0}[text])
 in text2
```

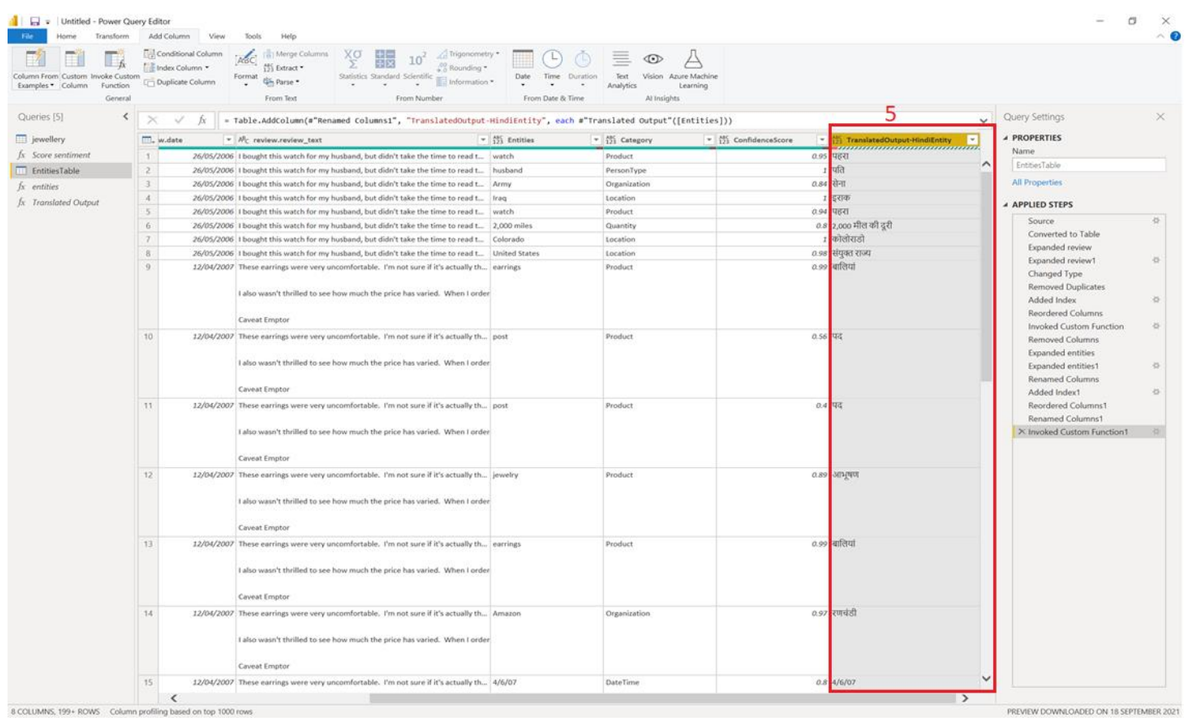
Important :

a. Make sure to add your Text Translation API key within quotes for apikey value. For this, use the Translator resource you created during Azure Translator workshop. Create a new Translator resource if you deleted that one.
b. In Translator workshop, we also created a custom model using Dictionary Data. Make sure to replace the Category id as highlighted with your custom ID. In case you did not create a custom model, we encourage you to go back to the Translator workshop to create one.
   Else, remove the underlined Category parameter & value from the endpoint, to use pre-built translation. However, do note that this might not give you the best results.

6. Make sure there are no syntax errors
7. Click Done







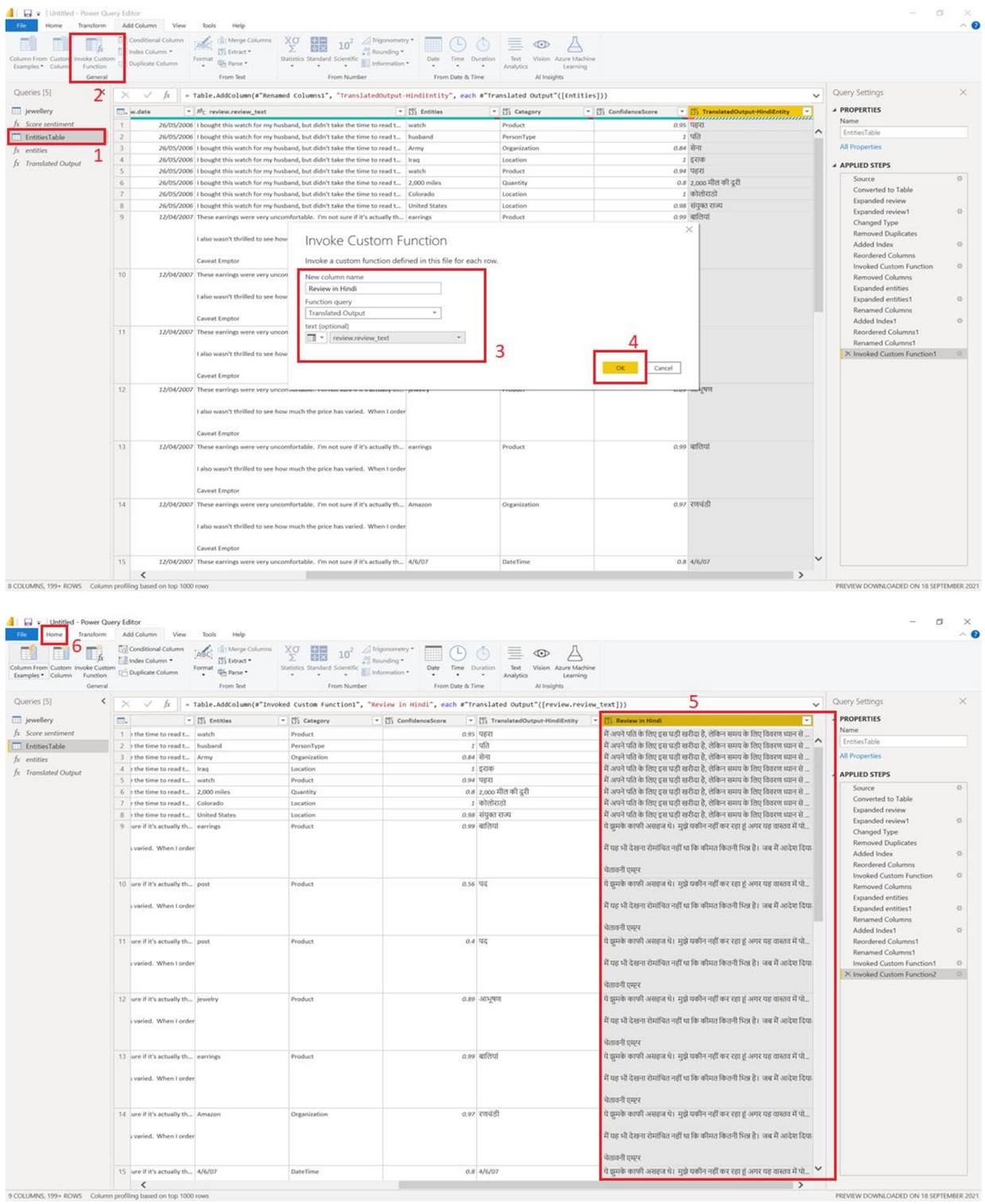
Invoke Custom function to translate entities to Hindi

We will now invoke the custom function we wrote to translate the entities to Hindi.

1. Select the Entities table
2. Select Invoke Custom Function

3. New column name : TranslatedOutput - HindiEntity
   Function query : Translated Output [From the dropdown select the name of the query you created above]
   Text : Entities [This is the column that contains the entities we extracted earlier]
4. Click Ok. Wait while the query is being executed.

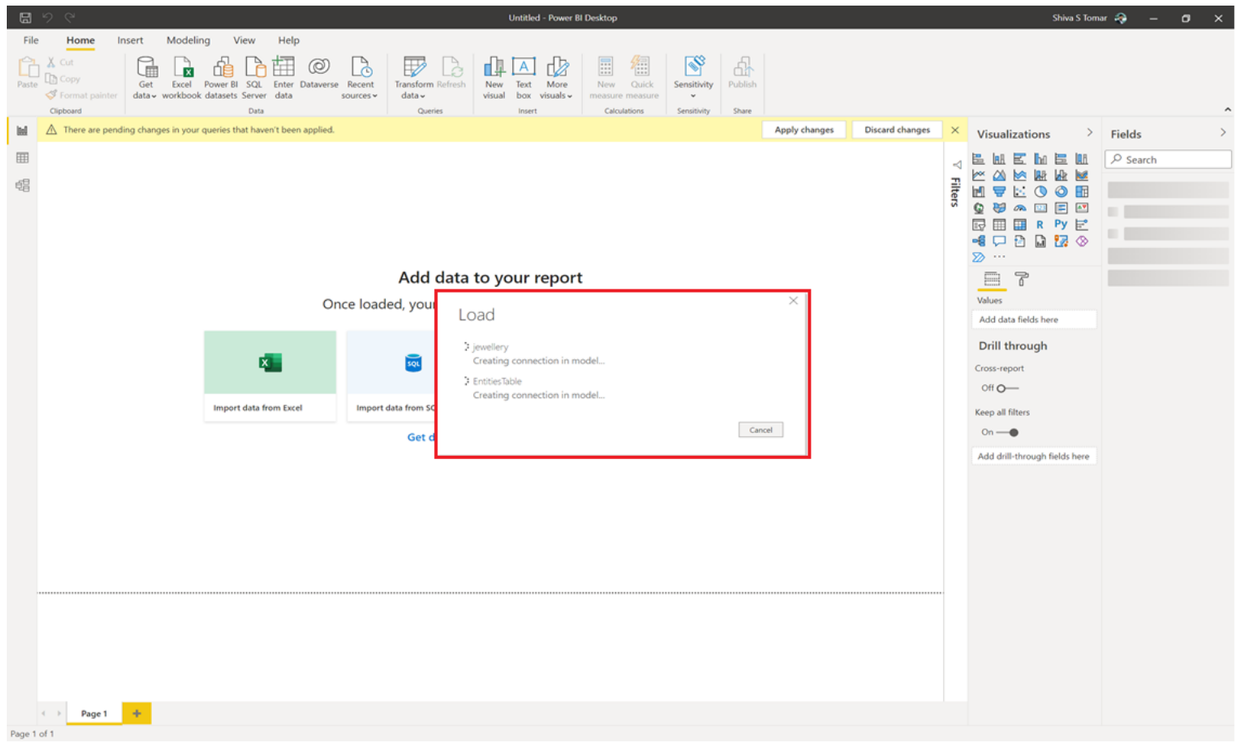
5. Observe the new column that is created with the entities name in Hindi.

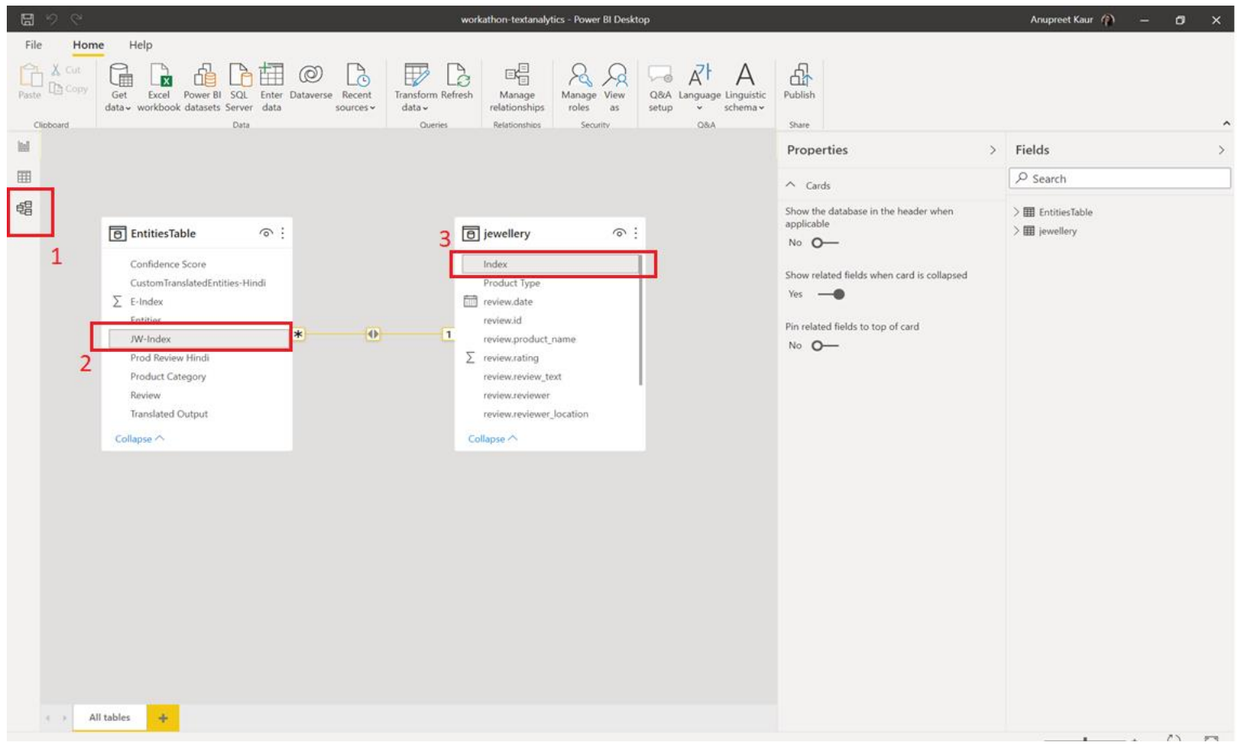


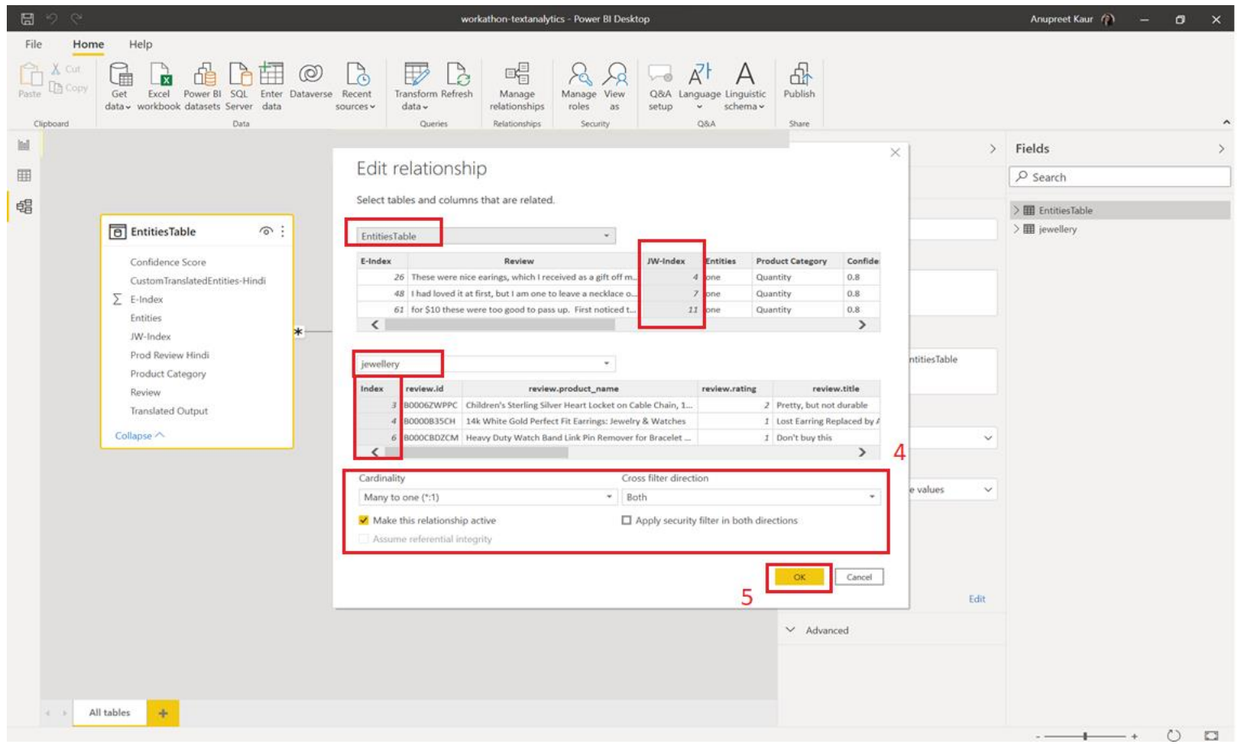
Invoke Custom function to translate customer reviews to Hindi

We will again invoke the translator function to translate the customer reviews to Hindi.

1. Select the Entities table
2. Select Invoke Custom Function
3. New column name : Review in Hindi
   Function query : Translated Output [From the dropdown select the name of the query you created above]
   Text : review.review_text [This is the column that contains the customer reviews]
4. Click Ok. Wait while the query is being executed.
5. Observe the new column that is created with the customer review text in Hindi.
6. Go to Home tab in the query editor and select Save & Close.

	Wait while Power BI applies the changes to both the datasets. This will take a while.





Create relational model

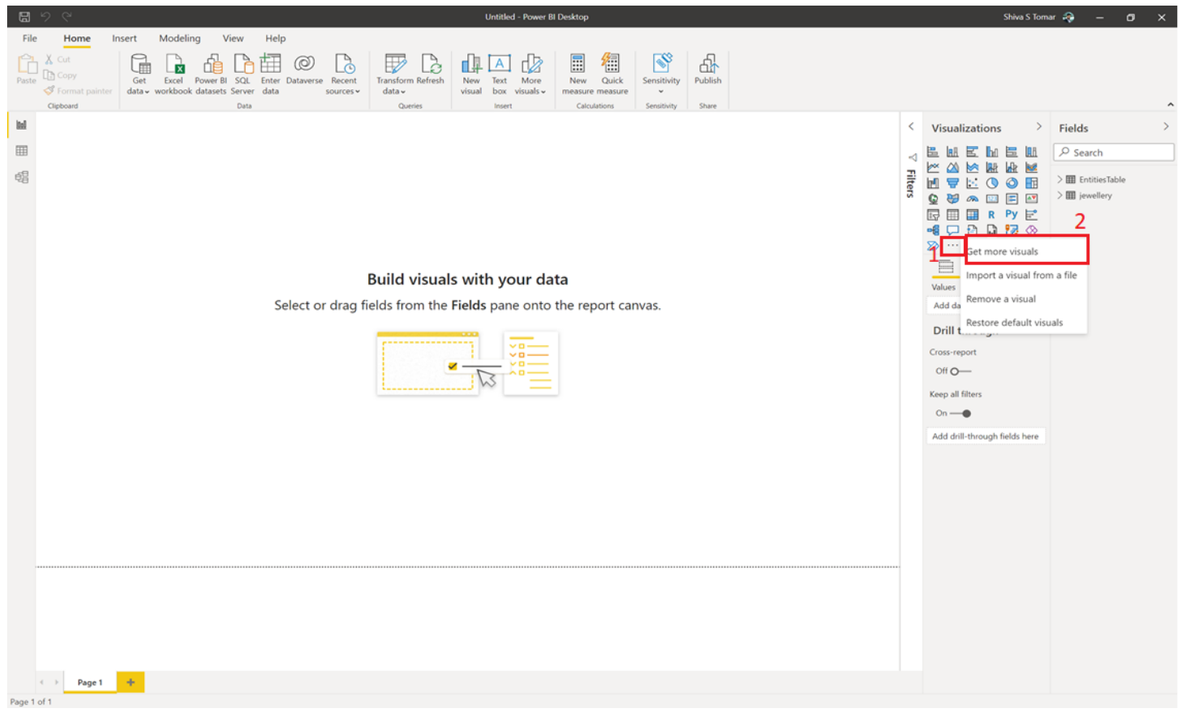
We will now be building the relationship between the 2 datasets, since we'll be creating visuals using both the datasets and we want to be able to filter the reports basis both the datasets collectively.

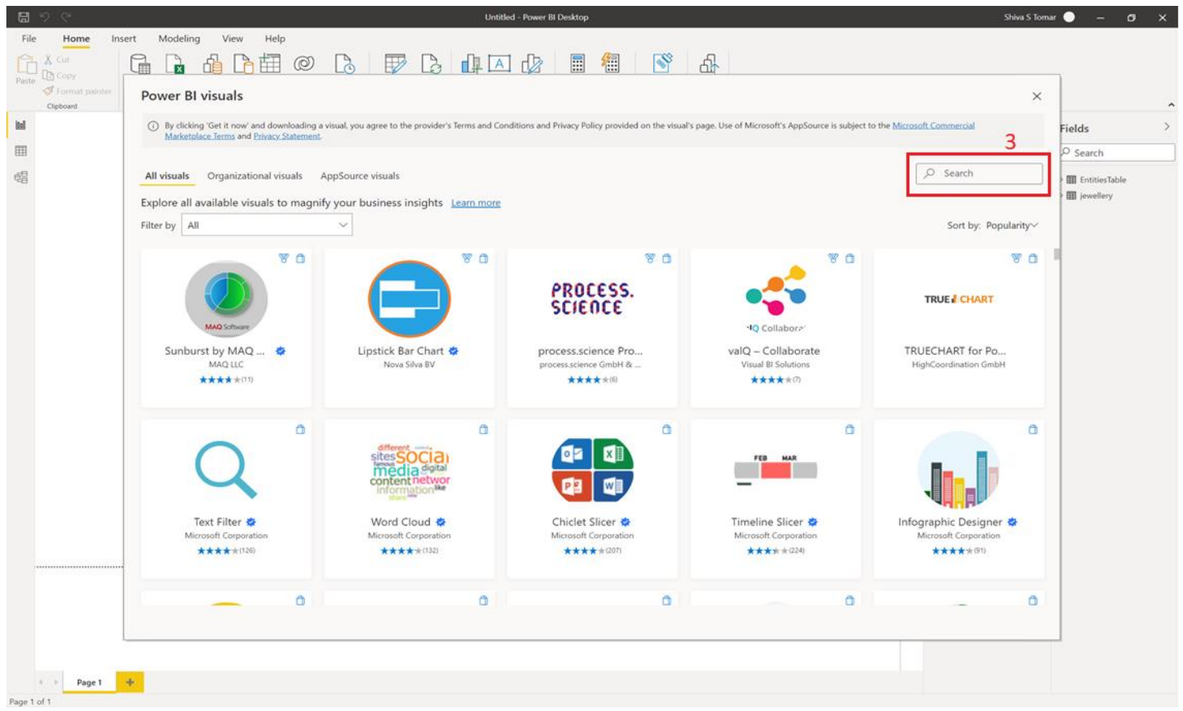
We will be building a bidirectional Many to One relationship from Entities table to Jewellery table.

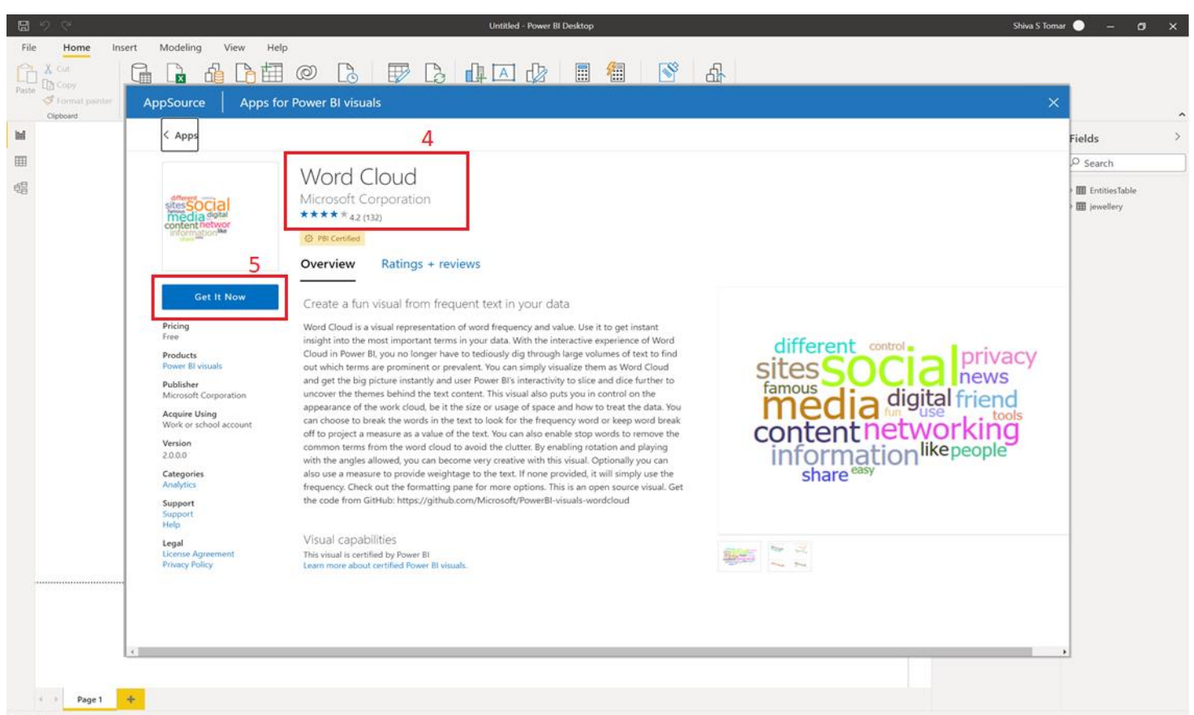
1. Go to the Models tab
2. Drag 'JW-Index' column in Entities table on top on 'Index' column in Jewellery table
3. Click on the relationship line that comes up

4. In the Edit relationship dialog box, make sure the columns from the respective tables are selected as highlighted. Make the following changes :
   Cardinality : Many to one (*.1) [Here, Entities is the first table, this is a many to one relationship from Entities table to Jewellery table]
   Cross filter direction : Both
   Make the relationship active should be checked

5. Click OK







Create Report

We will now start adding visuals to create a report for the customer review data.

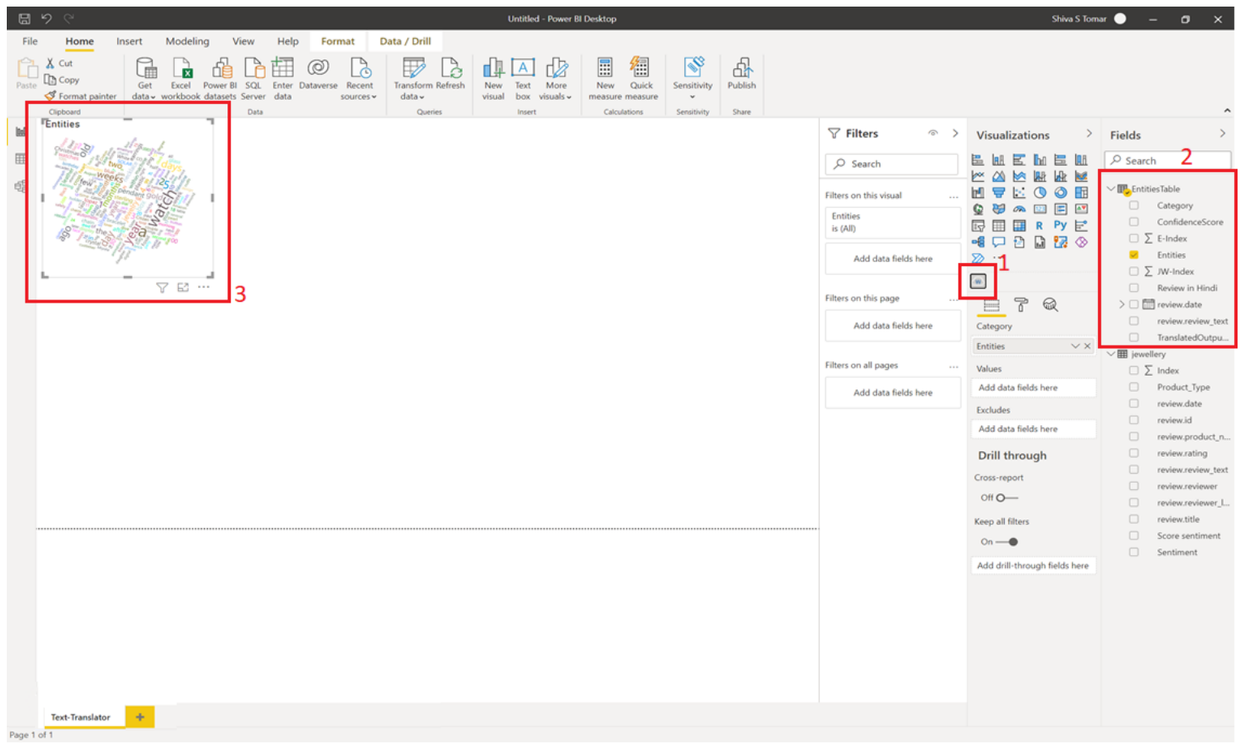
Import Word Cloud Visual

Since word cloud is not available as an out of the box visual in Power BI, we will be importing it.

1. Click the 3 dots icon at the end of the visual options
2. Select Get more visuals

3. Seach for ‘word cloud’

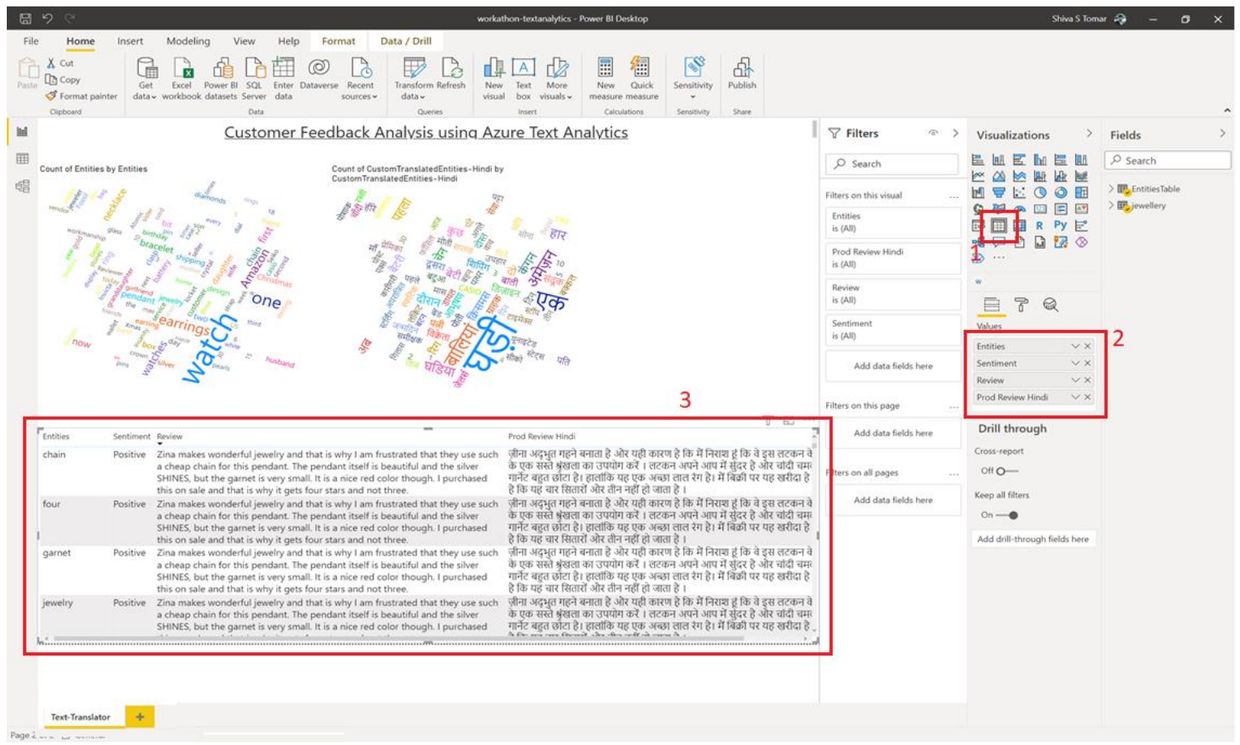
4. Open the Word Cloud visual
5. Click Get it Now



Create Word Cloud for extracted entities

1. Select the imported Word Cloud visual or drag it to the report canvas
2. Select Entities from Entities Table
3. The visual will appear as highlighted

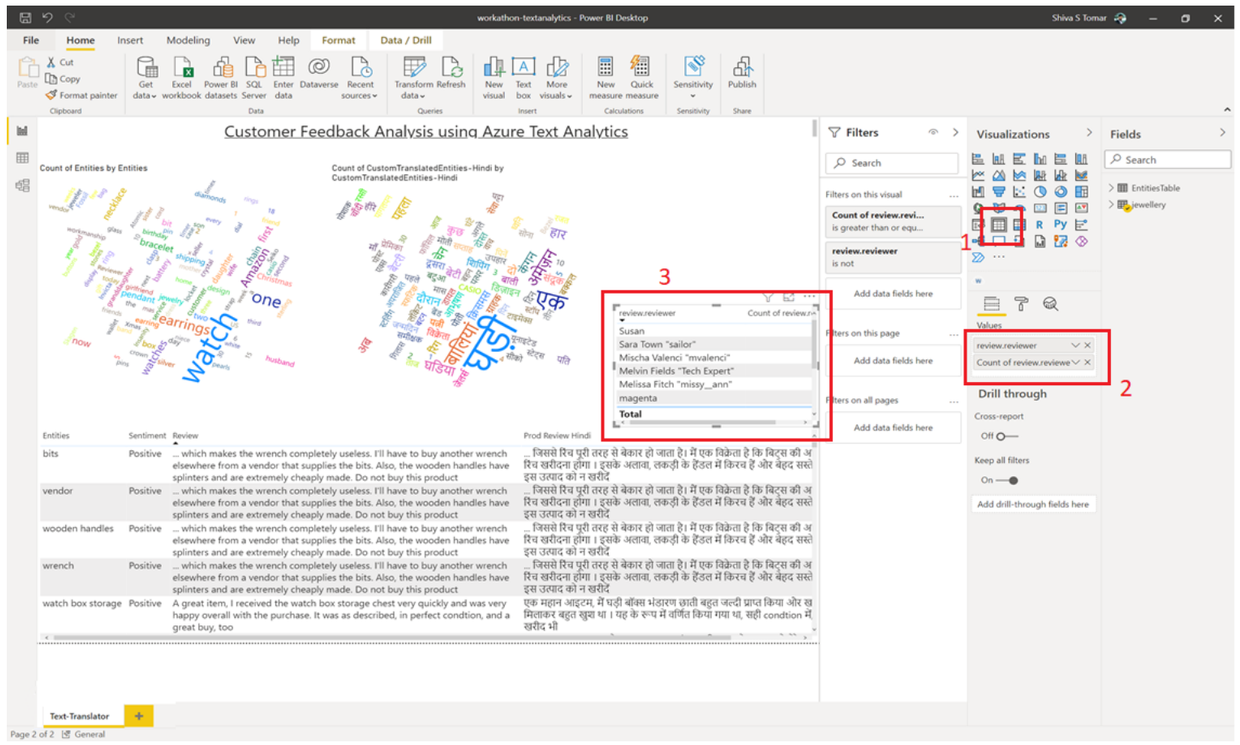
Similarly, create another Word Cloud for Entities in Hindi.



Create Review and Sentiment table

We will now create a visual that displays the Entities, sentiment and corresponding reviews in English and Hindi.

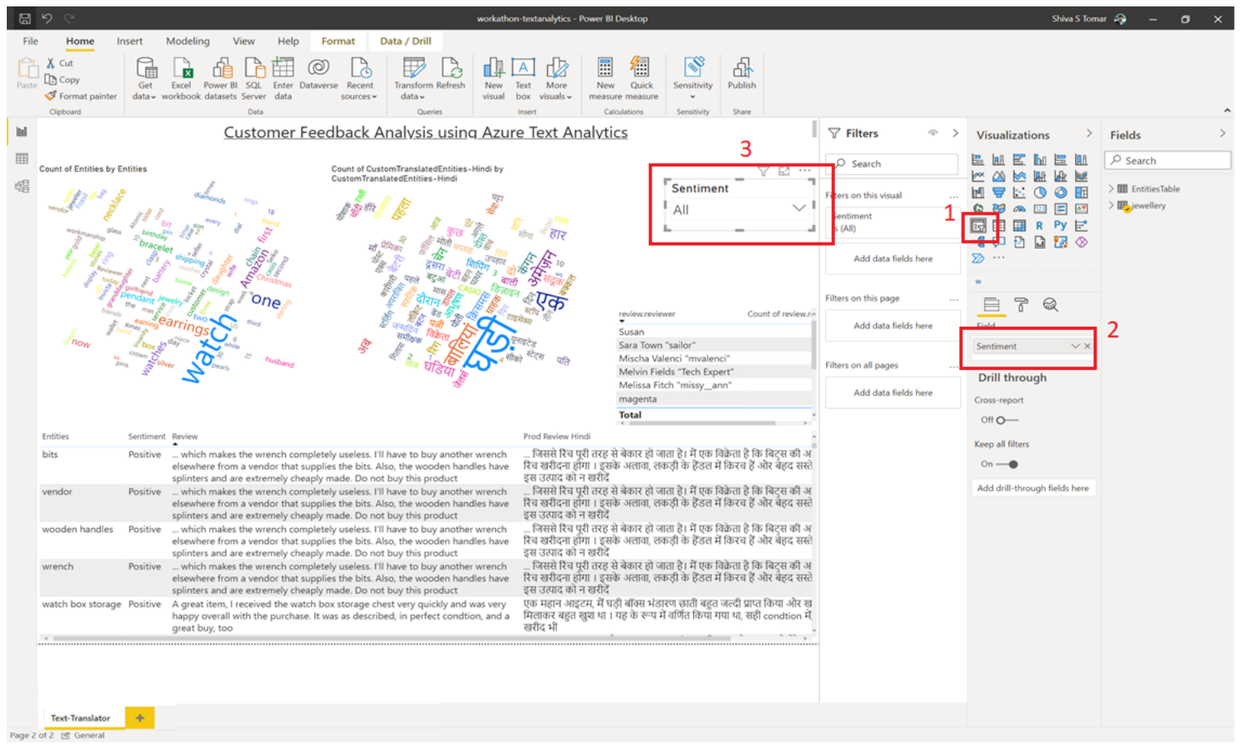
1. Select the Table visual or drag it to the report canvas
2. Select the columns from the respective tables as highlighted (Entities, reviews from entities table, sentiment from Jewellery table)
3. The visual will appear as highlighted



Create Customers' table

We will now create a visual that displays the Count of the customers.

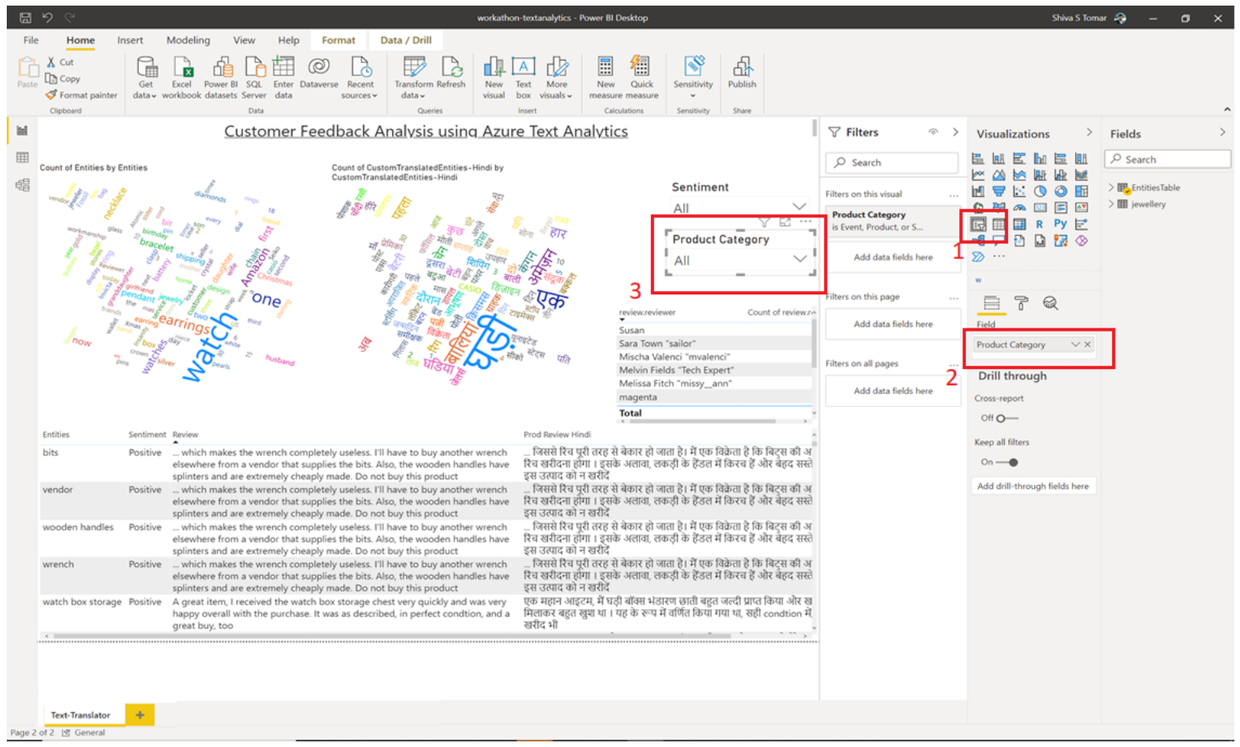
1. Select the Table visual or drag it to the report canvas
2. Set the columns as highlighted
3. The visual will appear as highlighted
4. You can additionally add more filters to the visual, using the 'Filters on this visual' option



Create Sentiment Filter

We will now create a filter that will allow us to filter the report by Customer Sentiment.

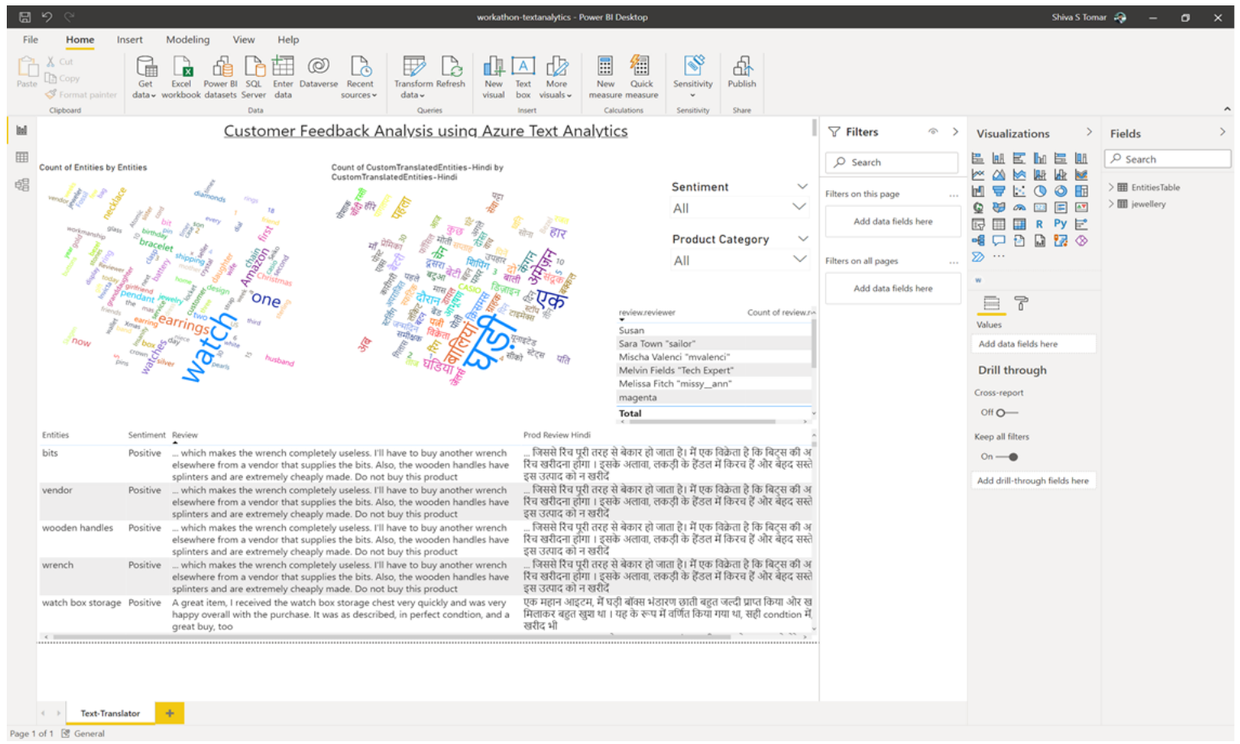
1. Select the Slicer visual or drag it to the report canvas
2. Set the Sentiment column from Jewellery table
3. The filter will appear as highlighted



Create Product Category Filter

We will now create a filter that will allow us to filter the report by Product Category, that we extracted as part of entity extraction process.

1. Select the Slicer visual or drag it to the report canvas
2. Set the Product Category column from Entities table
3. The filter will appear as highlighted



Once the report is ready, it will look something like this!

Homework

1. Create more visuals that suit the dataset
2. Create similar reports on other datasets you might have
3. Create new custom functions the way we created Entity extraction and Text Analytics queries

Additional recommended resources

Power BI documentation

Power BI Security best practices

Power BI video tutorials

For the additional resources on Text Analytics & Translator services, refer the respective workshops.