4.3 - Create a knowledge store with Azure AI Search

Overview

Persist the output from an Azure AI Search enrichment pipeline for independent analysis or downstream processing.

In this module you'll learn how to:

- Create a knowledge store from an Azure AI Search pipeline
- View data in projections in a knowledge store

Introduction

Azure AI Search enables you to create search solutions in which a pipeline of AI skills is used to enrich data and populate an index. The data enrichments performed by the skills in the pipeline supplement the source data with insights such as:

- The language in which a document is written.
- Key phrases that might help determine the main themes or topics discussed in a document.
- A sentiment score that quantifies how positive or negative a document is.
- Specific locations, people, organizations, or landmarks mentioned in the content.

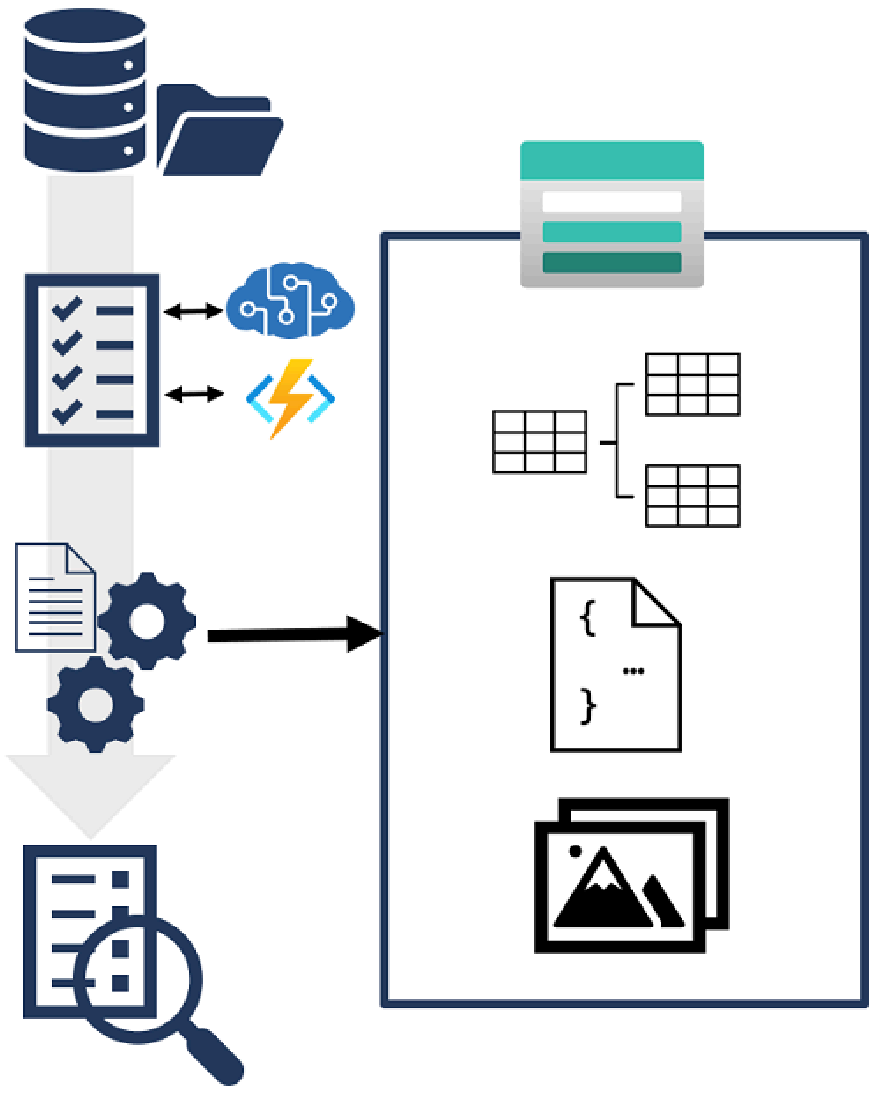
- AI-generated descriptions of images, or image text extracted by optical character recognition (OCR).

The enriched data in the index makes it possible to create a comprehensive search solution that goes beyond basic full text search of the source content.

Knowledge stores

While the index might be considered the primary output from an indexing process, the enriched data it contains might also be useful in other ways. For example:

- Since the index is essentially a collection of JSON objects, each representing an indexed record, it might be useful to export the objects as JSON files for integration into a data orchestration process using tools such as Azure Data Factory.
- You may want to **normalize the index records into a relational schema of tables** for analysis and reporting with tools such as Microsoft Power BI.
- Having extracted embedded images from documents during the indexing process, you might want to **save those images as files**.



Azure AI Search supports these scenarios by enabling you to define a *knowledge store* in the skillset that encapsulates your enrichment pipeline. The knowledge store consists of *projections* of the enriched data, which can be JSON objects, tables, or image files.

When an indexer runs the pipeline to create or update an index, the projections are generated and persisted in the knowledge store.

In this module, you'll implement a knowledge store for *Margie's Travel*, a fictitious travel agency that uses information in brochures and hotel reviews to help customers plan trips and you will learn how to:

- Create a knowledge store from an Azure AI Search pipeline
- View data in projections in a knowledge store

> Note: This module assumes you already know how to create and use an Azure AI Search solution that includes built-in skills. If not, complete the Create an Azure AI Search solution module first.

Define projections

The projections of data to be stored in your knowledge store are based on the document structures generated by the enrichment pipeline in your indexing process. Each skill in your skillset iteratively builds a JSON representation of the enriched data for the documents being indexed, and you can persist some or all of the fields in the document as projections.

Using the *Shaper* skill

The process of indexing incrementally creates a complex document that contains the various output fields from the skills in the skillset. This can result in a schema that is difficult to work with, and which includes collections of primitive data values that don't map easily to well-formed JSON.

To simplify the mapping of these field values to projections in a knowledge store, it's common to use the **Shaper** skill to create a new, field containing a **simpler structure for the fields** you want to map to projections.

For example, consider the following Shaper skill definition:

```
{
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
  "name": "define-projection",
  "description": "Prepare projection fields",
  "context": "/document",
  "inputs": [
    {
      "name": "file_name",
      "source": "/document/metadata_content_name"
    },
    {
      "name": "url",
      "source": "/document/url"
    },
    {
      "name": "sentiment",
      "source": "/document/sentimentScore"
    },
    {
      "name": "key_phrases",
      "source": null,
      "sourceContext": "/document/merged_content/keyphrases/*",
      "inputs": [
        {
          "name": "phrase",
          "source": "/document/merged_content/keyphrases/*"
        }
      ]
    }
  ],
  "outputs": [
    {
      "name": "output",
      "targetName": "projection"
    }
  ]
}
```

This Shaper skill creates a **projection** field with the following structure:

```
{
    "file_name": "file_name.pdf",
    "url": "https://<storage_path>/file_name.pdf",
    "sentiment": 1.0,
    "key_phrases": [
        {
            "phrase": "first key phrase"
        },
        {
            "phrase": "second key phrase"
        },
        {
            "phrase": "third key phrase"
        },
        ...
    ]
}
```

The resulting JSON document is well-formed, and easier to map to a projection in a knowledge store than the more complex document that has been built iteratively by the previous skills in the enrichment pipeline.

> Supplement information from ChatGPT: A projection is a specific way of organizing and saving data from the indexing process. It's like creating different views or formats of the enriched data based on what you need.
>
> For example:
>
> - **JSON Projection:** Saving the data as JSON files.
> - **Table Projection:** Organizing the data into tables.
> - **Image Projection:** Storing any images found in the data as separate image files.
>
>   Projections help you get the enriched data in the format that's most useful for your purposes.

Define a knowledge store

To define the knowledge store and the projections you want to create in it, you must create a **knowledgeStore** object in the skillset that specifies the Azure Storage connection string for the storage account where you want to create projections, and the definitions of the projections themselves.

You can define object projections, table projections, and file projections depending on what you want to store; however note that you must define a separate *projection* for each type of projection, even though each projection contains lists for tables, objects, and files.

Projection types are mutually exclusive in a projection definition, so only one of the projection type lists can be populated. If you create all three kinds of projection, you must include a projection for each type; as shown here:

```
"knowledgeStore": {
      "storageConnectionString": "<storage_connection_string>",
      "projections": [
        {
            "objects": [
                {
                "storageContainer": "<container>",
                "source": "/projection"
                }
            ],
            "tables": [],
            "files": []
        },
        {
            "objects": [],
            "tables": [
                {
                "tableName": "KeyPhrases",
                "generatedKeyName": "keyphrase_id",
                "source": "projection/key_phrases/*",
                },
                {
                "tableName": "docs",
                "generatedKeyName": "document_id",
                "source": "/projection"
                }
            ],
            "files": []
        },
        {
            "objects": [],
            "tables": [],
            "files": [
                {
                "storageContainer": "<container>",
                "source": "/document/normalized_images/*"
                }
            ]
        }
```

```
    ]
}
```

For **object** and **file** projections, the specified container will be created if it does not already exist. An Azure Storage table will be created for each **table** projection, with the mapped fields and a unique key field with the name specified in the **generatedKeyName** property. These key fields can be used to define relational joins between the tables for analysis and reporting.

Exercise - Create a knowledge store

Azure AI Search uses an enrichment pipeline of AI skills to extract AI-generated fields from documents and include them in a search index. While the index might be considered the primary output from an indexing process, the enriched data it contains might also be useful in other ways. For example:

- Since the index is essentially a collection of JSON objects, each representing an indexed record, it might be useful to export the objects as JSON files for integration into a data orchestration process using tools such as Azure Data Factory.
- You may want to normalize the index records into a relational schema of tables for analysis and reporting with tools such as Microsoft Power BI.
- Having extracted embedded images from documents during the indexing process, you might want to save those images as files.

In this exercise, you'll implement a knowledge store for *Margie's Travel*, a fictitious travel agency that uses information in brochures and hotel reviews to help customers plan trips.

Prepare to develop an app in Visual Studio Code

You'll develop your search app using Visual Studio Code. The code files for your app have been provided in a GitHub repo.

> **Tip**: If you have already cloned the **mslearn-knowledge-mining** repo, open it in Visual Studio code. Otherwise, follow these steps to clone it to your development environment.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/mslearn-knowledge-mining` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo.

    > **Note**: If you are prompted to add required assets to build and debug, select **Not Now**.

Create Azure resources

> **Note**: If you have previously completed the [Create an Azure AI Search solution]() exercise, and still have these Azure resources in your subscription, you can skip this section and start at the **Create a search solution** section. Otherwise, follow the steps below to provision the required Azure resources.

1. In a web browser, open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. View the **Resource groups** in your subscription.
3. If you are using a restricted subscription in which a resource group has been provided for you, select the resource group to view its properties. Otherwise, create a new resource group with a name of your choice, and go to it when it has been created.
4. On the **Overview** page for your resource group, note the **Subscription ID** and **Location**. You will need these values, along with the name of the resource group in subsequent steps.
5. In Visual Studio Code, expand the **Labfiles/03-knowledge-store** folder and select **setup.cmd**. You will use this batch script to run the Azure command line interface (CLI) commands required to create the Azure resources you need.
6. Right-click the the **03-knowledge-store** folder and select **Open in Integrated Terminal**.
7. In the terminal pane, enter the following command to establish an authenticated connection to your Azure subscription.
   ```
   az login --output none
   ```
8. When prompted, sign into your Azure subscription. Then return to Visual Studio Code and wait for the sign-in process to complete.
9. Run the following command to list Azure locations.
   ```
   az account list-locations -o table
   ```
10. In the output, find the **Name** value that corresponds with the location of your resource group (for example, for *East US* the corresponding name is *eastus*).
11. In the **setup.cmd** script, modify the **subscription_id**, **resource_group**, and **location** variable declarations with the appropriate values for your subscription ID, resource group name, and location name. Then save your changes.
12. In the terminal for the **03-knowledge-store** folder, enter the following command to run the script:
    ```
    ./setup
    ```

    > **Note**: The Search CLI module is in preview, and may get stuck in the *- Running ..* process. If this happens for over 2 minutes, press CTRL+C to cancel the long-running operation, and then select **N** when asked if you want to terminate the script. It should then complete successfully.
    >
    > If the script fails, ensure you saved it with the correct variable names and try again.

13. When the script completes, review the output it displays and note the following information about your Azure resources (you will need these values later):
    - Storage account name
    - Storage connection string
    - Azure AI Services account
    - Azure AI Services key
    - Search service endpoint
    - Search service admin key
    - Search service query key
14. In the Azure portal, refresh the resource group and verify that it contains the Azure Storage account, Azure AI Services resource, and Azure AI Search resource.

Create a search solution

Now that you have the necessary Azure resources, you can create a search solution that consists of the following components:

- A **data source** that references the documents in your Azure storage container.
- A **skillset** that defines an enrichment pipeline of skills to extract AI-generated fields from the documents. The skillset also defines the *projections* that will be generated in your *knowledge store*.
- An **index** that defines a searchable set of document records.
- An **indexer** that extracts the documents from the data source, applies the skillset, and populates the index. The process of indexing also persists the projections defined in the skillset in the knowledge store.

In this exercise, you'll use the Azure AI Search REST interface to create these components by submitting JSON requests.

Prepare JSON for REST operations

You'll use the REST interface to submit JSON definitions for your Azure AI Search components.

1. In Visual Studio Code, in the **03-knowledge-store** folder, expand the **create-search** folder and select **data_source.json** (data_source). This file contains a JSON definition for a data source named **margies-knowledge-data**.
2. Replace the **YOUR_CONNECTION_STRING** placeholder with the connection string for your Azure storage account, which should resemble the following:

   ```
   DefaultEndpointsProtocol=https;AccountName=ai102str123;AccountKey=12345abcdefg...==;
   EndpointSuffix=core.windows.net
   ```

   You can find the connection string on the **Access keys** page for your storage account in the Azure portal.
3. Save and close the updated JSON file.
4. In the **create-search** folder, open **skillset.json**. This file contains a JSON definition for a skillset named **margies-knowledge-skillset**.
5. At the top of the skillset definition, in the **cognitiveServices** element, replace the **YOUR_COGNITIVE_SERVICES_KEY** placeholder with either of the keys for your Azure AI Services resources.

   *You can find the keys on the **Keys and Endpoint** page for your Azure AI Services resource in the Azure portal.*
6. At the end of the collection of skills in your skillset, find the **Microsoft.Skills.Util.ShaperSkill** skill named **define-projection**. This skill defines a JSON structure for the enriched data that will be used for the projections that the pipeline will persist on the knowledge store for each document processed by the indexer.
7. At the bottom of the skillset file, observe that the skillset also includes a **knowledgeStore** definition, which includes a connection string for the Azure Storage account where the knowledge store is to be created, and a collection of **projections**. This skillset includes three *projection groups*:
   - A group containing an *object* projection based on the **knowledge_projection** output of the shaper skill in the skillset.
   - A group containing a *file* projection based on the **normalized_images** collection of image data extracted from the documents.
   - A group containing the following *table* projections:
     - **KeyPhrases**: Contains an automatically generated key column and a **keyPhrase** column mapped to the **knowledge_projection/key_phrases/** collection output of the shaper skill.

- **Locations**: Contains an automatically generated key column and a **location** column mapped to the **knowledge_projection/key_phrases/** collection output of the shaper skill.
- **ImageTags**: Contains an automatically generated key column and a **tag** column mapped to the **knowledge_projection/image_tags/** collection output of the shaper skill.
- **Docs**: Contains an automatically generated key column and all of the **knowledge_projection** output values from the shaper skill that are not already assigned to a table.

8. Replace the **YOUR_CONNECTION_STRING** placeholder for the **storageConnectionString** value with the connection string for your storage account.
9. Save and close the updated JSON file.
10. In the **create-search** folder, open **index.json**. This file contains a JSON definition for an index named **margies-knowledge-index**.
11. Review the JSON for the index, then close the file without making any changes.
12. In the **create-search** folder, open **indexer.json**. This file contains a JSON definition for an indexer named **margies-knowledge-indexer**.
13. Review the JSON for the indexer, then close the file without making any changes.

Submit REST requests

Now that you've prepared the JSON objects that define your search solution components, you can submit the JSON documents to the REST interface to create them.

1. In the **create-search** folder, open **create-search.cmd**. This batch script uses the cURL utility to submit the JSON definitions to the REST interface for your Azure AI Search resource.
2. Replace the **YOUR_SEARCH_URL** and **YOUR_ADMIN_KEY** variable placeholders with the **Url** and one of the **admin keys** for your Azure AI Search resource.
   *You can find these values on the **Overview** and **Keys** pages for your Azure AI Search resource in the Azure portal.*
3. Save the updated batch file.
4. Right-click the the **create-search** folder and select **Open in Integrated Terminal**.
5. In the terminal pane for the **create-search** folder, enter the following command run the batch script.
   `./create-search`
6. When the script completes, in the Azure portal, on the page for your Azure AI Search resource, select the **Indexers** page and wait for the indexing process to complete.
   *You can select **Refresh** to track the progress of the indexing operation. It may take a minute or so to complete.*

   > **Tip**: If the script fails, check the placeholders you added in the **data_source.json** and **skillset.json** files as well as the **create-search.cmd** file. After correcting any mistakes, you may need to use the Azure portal user interface to delete any components that were created in your search resource before re-running the script.

View the knowledge store

After you have run an indexer that uses a skillset to create a knowledge store, the enriched data extracted by the indexing process is persisted in the knowledge store projections.

View object projections

The *object* projections defined in the Margie's Travel skillset consist of a **JSON** file for each indexed document. These files are stored in a blob container in the Azure Storage account specified in the skillset definition.

1. In the Azure portal, view the Azure Storage account you created previously.
2. Select the **Storage browser** tab (in the pane on the left) to view the storage account in the storage explorer interface in the Azure portal.
3. Expand **Blob containers** to view the containers in the storage account. In addition to the **margies** container where the source data is stored, there should be two new containers: **margies-images** and **margies-knowledge**. These were created by the indexing process.
4. Select the **margies-knowledge** container. It should contain **a folder for each indexed document**.
5. Open any of the folders, and then download and open the **knowledge-projection.json** file it contains. Each JSON file contains a representation of an indexed document, including the enriched data extracted by the skillset as shown here.

```
{"file_id":"abcd1234....", "file_name":"Margies Travel Company Info.pdf",
"url":"https://store....blob.core.windows.net/margies/...pdf", "language":"en",
"sentiment":0.83164644241333008, "key_phrases":["Margie’s Travel", "Margie's Travel", "best
travel experts", "world-leading travel agency", "international reach"], "locations":
["Dubai", "Las Vegas", "London", "New York", "San Francisco"], "image_tags":["outdoor",
"tree", "plant", "palm"] }
```

The ability to create *object* projections like this enables you to generate enriched data objects that can be incorporated into an enterprise data analysis solution - for example by ingesting the JSON files into an Azure Data Factory pipeline for further processing or loading into a data warehouse.

View file projections

The *file* projections defined in the skillset create **JPEG** files for each image that was extracted from the documents during the indexing process.

1. In the *Storage browser* interface in the Azure portal, select the **margies-images** blob container. This container contains a folder for each document that contained images.
2. Open any of the folders and view its contents - each folder contains at least one *.jpg file.
3. Open any of the image files to verify that they contain images extracted from the documents.

The ability to generate *file* projections like this **makes indexing an efficient way to extract embedded images from a large volume of documents.**

View table projections

The *table* projections defined in the skillset form a **relational schema of enriched data**.

1. In the *Storage browser* interface in the Azure portal, expand **Tables**.
2. Select the **docs** table to view its columns. The columns include some standard Azure Storage table columns - to hide these, modify the **Column Options** to select only the following columns:
   - **document_id** (the key column automatically generated by the indexing process)
   - **file_id** (the encoded file URL)
   - **file_name** (the file name extracted from the document metadata)
   - **language** (the language in which the document is written)
   - **sentiment** the sentiment score calculated for the document.

- **url** the URL for the document blob in Azure storage.

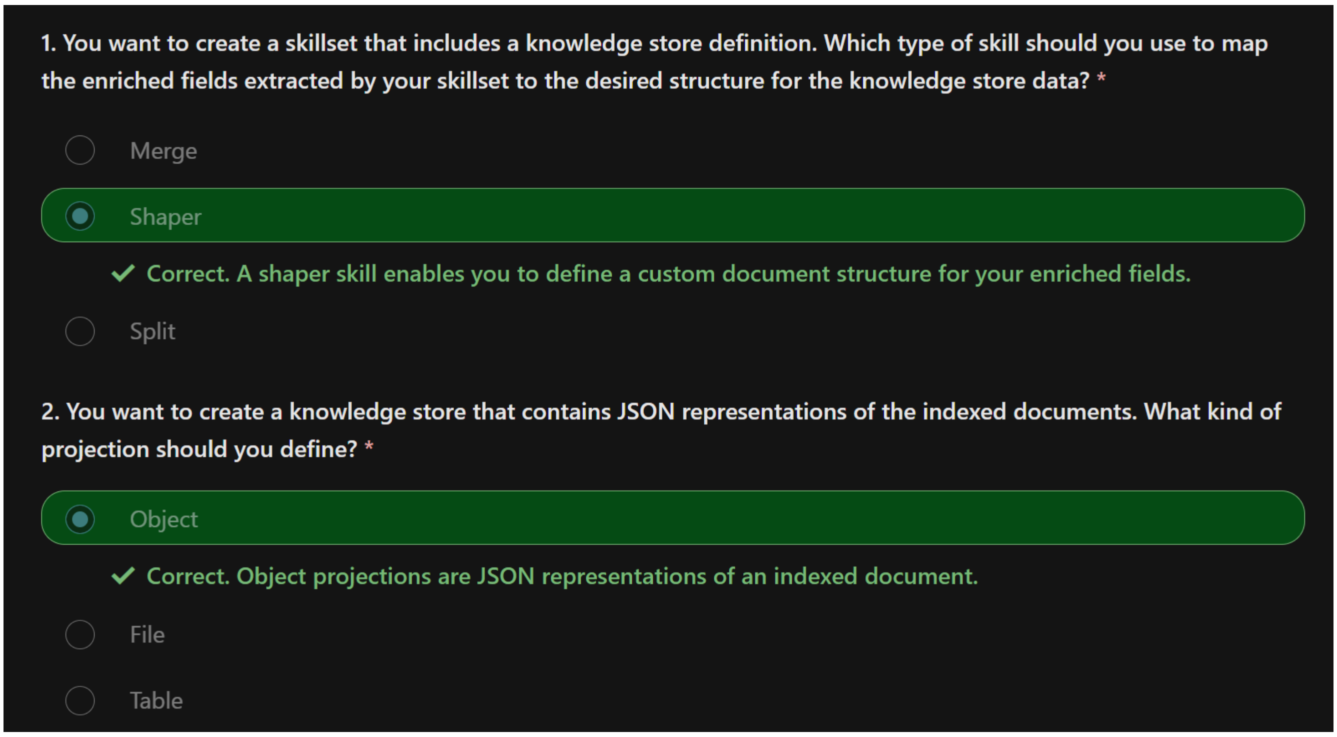
3. View the other tables that were created by the indexing process:
    - **ImageTags** (contains a row for each individual image tag with the **document_id** for the document in which the tag appears).
    - **KeyPhrases** (contains a row for each individual key phrase with the **document_id** for the document in which the phrase appears).
    - **Locations** (contains a row for each individual location with the **document_id** for the document in which the location appears).

The ability to create *table* projections enables you to build analytical and reporting solutions that query the relational schema; for example, using Microsoft Power BI. The automatically generated key columns can be used to join the tables in queries - for example to return all of the locations mentioned in a specific document.

More information

To learn more about creating knowledge stores with Azure AI Search, see the Azure AI Search documentation.

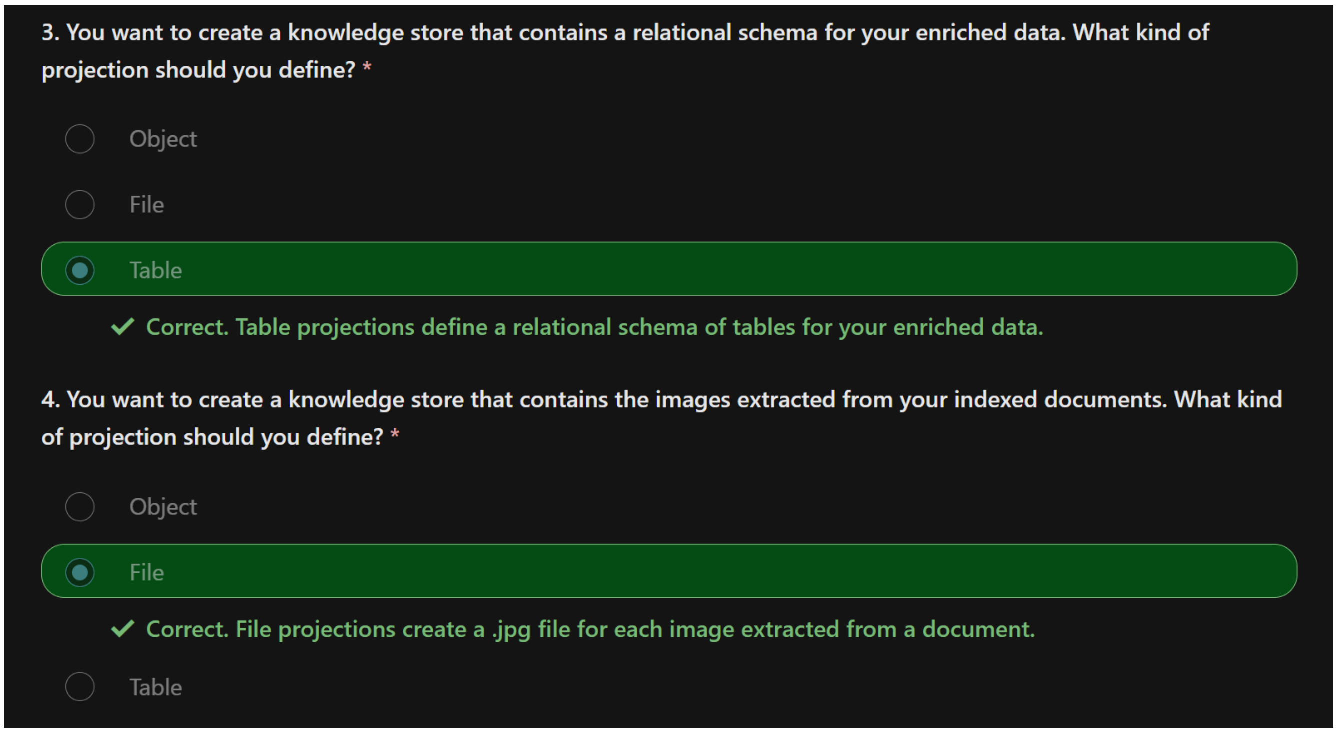
Knowledge Check


1. You want to create a skillset that includes a knowledge store definition. Which type of skill should you use to map the enriched fields extracted by your skillset to the desired structure for the knowledge store data? *

- Merge
- [✓ Shaper]
  ✓ Correct. A shaper skill enables you to define a custom document structure for your enriched fields.
- Split

2. You want to create a knowledge store that contains JSON representations of the indexed documents. What kind of projection should you define? *

- [✓ Object]
  ✓ Correct. Object projections are JSON representations of an indexed document.
- File
- Table


3. You want to create a knowledge store that contains a relational schema for your enriched data. What kind of projection should you define? *

- Object
- File
- [✓ Table]
  ✓ Correct. Table projections define a relational schema of tables for your enriched data.

4. You want to create a knowledge store that contains the images extracted from your indexed documents. What kind of projection should you define? *

- Object
- [✓ File]
  ✓ Correct. File projections create a .jpg file for each image extracted from a document.
- Table

Summary

In this module, you learned how to use Azure AI Search to create a *knowledge store* in which to persist enriched data extracted from your data sources by the indexing process. Knowledge stores can contain data in three kinds of *projection*:

- ***Object*** projections are **JSON** representations of the indexed documents.
- ***File*** projections are JPEG **files** containing image data extracted from documents.
- ***Table*** projections create a **relational schema** for the extracted data.

For more information about Azure AI Search, take a look at the service documentation.

✍️ Compiled by Kenneth Leung (2025)