EISART-59755 : [Document the process for automation of normalized attribute extraction for new items](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fjira.kroger.com%2Fjira%2Fbrowse%2FEISART-59755&data=05%7C02%7Csophia.yue%40kroger.com%7C992ba64290a1451b13d208dc11241de1%7C8331e14a91344288bf5a5e2c8412f074%7C0%7C0%7C638404094073775055%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=hCevfzsTG1VnF%2Fnb%2FCVK7BBwgREfja49VHA1IRWJcOM%3D&reserved=0)

Normalization process: [Finding Nemo - Active Work - EIX - Enterprise Item Experience - Confluence (kroger.com)](https://confluence.kroger.com/confluence/display/EIS/Finding+Nemo+-+Active+Work)

[Data Science Workflow - EIX - Enterprise Item Experience - Confluence (kroger.com)](https://confluence.kroger.com/confluence/display/EIS/Data+Science+Workflow) Data Science workflow for normalized Attribute Extraction Stories

[[EISART-59784] Insert normalized data into PIM Production - Kroger Technology](https://jira.kroger.com/jira/browse/EISART-59784)

[Insert normalized data into PIM Production](https://jira.kroger.com/jira/browse/EISART-59784) – not related to 59755

**Description**

The next PIM Deployment will be on Jan 17th.

We should have a file prepared with all normalized attribute values ready to insert into PIM Production that mimics what we put into stage plus any additional products with added attributes since the stage insertion.

[[EISART-59757] COLOR: Create Code and Configs for Normalized Attribution - Kroger Technology](https://jira.kroger.com/jira/browse/EISART-59757)

Acceptance Criteria:

The shared Python code must be able to:

1) Accept Item Description and Department as input.

2) Utilize the description and department-specific configuration parameters to perform fuzzy match for extracting Color attribute value.

3)Pre-validated dataset to be used as benchmark for validating code outcomes.

#### Description

The goal is to develop code that can be used for normalized attribution of specific item categories in the future.

[[EISART-59759] SCENT: Create Code and Configs for Normalized Attribution - Kroger Technology](https://jira.kroger.com/jira/browse/EISART-59759)

SCENT: Create Code and Configs for Normalized Attribution

Acceptance Criteria:

The shared Python code must be able to:

1) Accept Item Description and Department as input.

2) Utilize the description and department-specific configuration parameters to perform fuzzy match for extracting Scent attribute value.

3)Pre-validated dataset to be used as benchmark for validating code outcomes.

#### Description

The goal is to develop code that can be used for normalized Scent attribution of specific item categories in the future.

[[EISART-59758] FLAVOR: Create Code and Configs for Normalized Attribution - Kroger Technology](https://jira.kroger.com/jira/browse/EISART-59758)

Acceptance Criteria:

The shared Python code must be able to:

1) Accept Item Description and Department as input.

2) Utilize the description and department-specific configuration parameters to perform fuzzy match for extracting Flavor attribute value.

3)Pre-validated dataset to be used as benchmark for validating code outcomes.

#### Description

The goal is to develop code that can be used for normalized Flavor attribution of specific item categories in the future.

[6:17 PM] Chakraborty, Shiladitya (NonEmp)

Sharing an outline of what the code is intended to do

1) Allow user to select a Department from drop -down list ( e.g. Kitchen & Dining)

  and  upload a input csv file containing item descriptions .

2) Load the correct config file for that department , attribute combination based on the filename

     ( e.g.  COLOR\_Kitchen & Dining\_SHILADITYA\_12082023.json)

3) From the config file, extract the following:

    a. Normalized list of Color values that Paul recommended to use for this dept.

       Example: ['B:ACK','BROWN', 'BLUE', DARK BLUE', 'MULTI-COLOR', 'MULTI COLOR']

      Note that this list may include multiple spelling variations of the same color.

      (e.g.  'MULTI-COLOR' &  'MULTI COLOR')

    b.  Substitutions:  Mapping of how to rename some of the colors in the final output. Example:   {'DARK BLUE': 'BLUE', 'MULTI COLOR': 'MULTI-COLOR'}

4) Run the 'color\_match' function by passing the descriptions in user's file and the above configs.

5) Save the output file.

NOTE: the output file will be  a copy of the input file with the added column 'Normalized\_Color'.

**Introduction**

* Document the process for automation of normalized attribute extraction for new items.
* Allow user to select a department from drop -down list ( e.g. Kitchen & Dining)
* Upload an input csv file containing item descriptions.
* Utilize the description and department-specific configuration parameters to perform fuzzy match for extracting Flavor attribute value.
* Save the output file with fields: GTIN\_NO, DESCRIPTION, Normalized\_Color.

[Automation of normalized attribute extraction for new items - EIX - Enterprise Item Experience - Confluence (kroger.com)](https://confluence.kroger.com/confluence/pages/resumedraft.action?draftId=427962940&draftShareId=f51d7578-ab8b-4bdc-93f9-b84db616c26e&)

Code cell title

Run this cell to display the item category selection drop-down, then make the selection

Pantry, Beverages, Candy -- FLAVOR

Beauty, Personal Care, Cleaning and Household – SCENT

Purpose of Configuration file

### Configuration file is a json file

### Location of the file is '/dbfs/FileStore/tables/DATA\_SCIENCE/FuzzyConfigs/'

Naming convention of the file is attrib\_cf\_dpt\_me\_today.json

Attrib: Uppercase of attribute, e.g., 'COLOR', FLAVOR, ‘SCENT’

cf\_dpt: cfic (customer facing) department, e.g., ‘Floral’, ‘Rug’,

me: Name of developer who created the config file for accountability purpose

today: mm\_dd\_ccyy from system date

Example: COLOR\_Floral\_SOPHIA\_01052024.json

Attrib: ‘COLOR’

cf\_dpt: ‘Floral’

me: ‘Sophia’

today: 01052024

upload input file

print('Upload the file containing item descriptions to be used for attribute extraction')

upload\_w = widgets.FileUpload(

accept='.csv', # Accepted file extension e.g. '.txt', '.pdf', 'image/\*', 'image/\*,.pdf'

multiple=False # True to accept multiple files upload else False

)

display(upload\_w)

attrib + '\_' +cf\_dpt + '\_' + me + '\_'+ today+ '.json

COLOR\_Floral\_SOPHIA\_01052024.json

CFIC\_DPT: CFIC department. The value could be 'Floral',

Attribute: Attribute. COLOR

Normalized\_list:  List of attribute values that Paul recommended to use for this dept.

Substitutions: Mapping of how to rename some of the colors in the final output. Example:   {'DARK BLUE': 'BLUE', 'MULTI COLOR': 'MULTI-COLOR'}

<https://adb-4812933386228410.10.azuredatabricks.net/files/tables/DATA_SCIENCE/FuzzyConfigs/FLAVOR_Candy_CONFIGS.json>

https://adb-4812933386228410.10.azuredatabricks.net/files/tables/DATA\_SCIENCE/FuzzyConfigs/SCENT\_PersonalCare\_CONFIGS.json

rename = {'Home Decor (excluding Rugs)':'Home\_Decor\_NoRugs', 'Home Decor (Rugs only)':'Home\_Decor\_Rugs'}

attr\_dpt\_map = {'Color':['Kitchen & Dining', 'Home Decor (excluding Rugs)', 'Home Decor (Rugs only)' ,'Floral'],\

                'Flavor': ['Pantry','Beverages','Ice Cream','Candy'],\

                 'Scent':['Cleaning & Household','Beauty','Personal Care']}

Example:

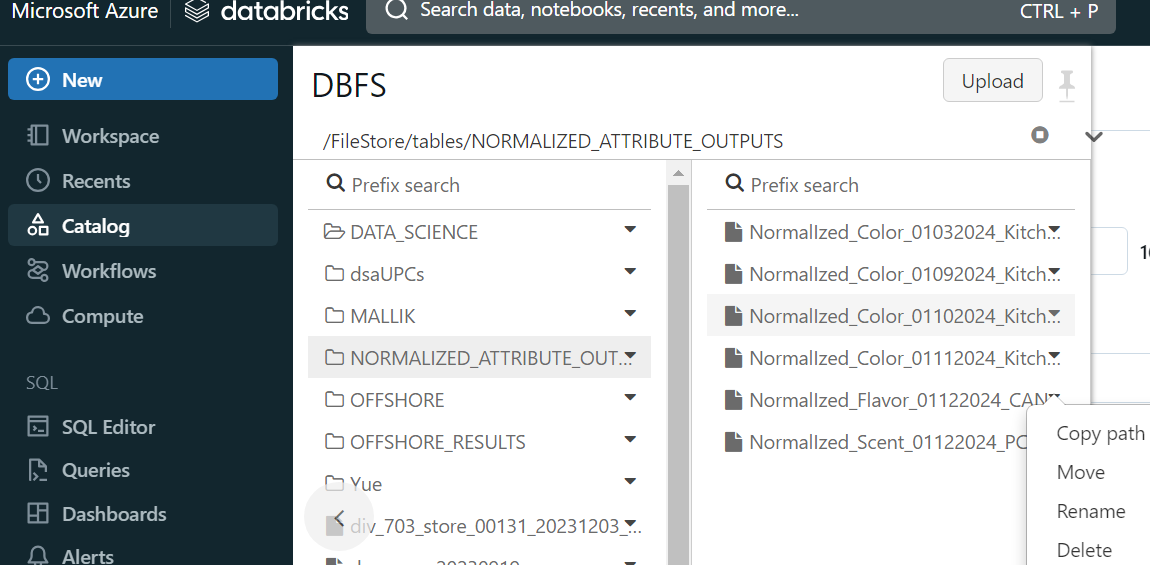
{"CFIC\_DPT": "Personal Care", "Attribute": "Scent", "Normalized\_list": ["OATMEAL", "CITRUS", "TURMERIC", "SPORT",…..], "REGULAR"], "Substitutions": {"CAFFEINE": "COFFEE", "BBQ": "BARBECUE",… , "CAFFEINE FREE": "OTHER"}}

Output path:

filed will be saved in: /dbfs/FileStore/tables/NORMALIZED\_ATTRIBUTE\_OUTPUTS

Filename format:

If the input file is 'XYZ.csv', the output file will be called 'NORMALIZED\_Color\_<Today's Date>\_XYZ.csv'



* Sing in Azure Databricks > Click 'Catalog' from the left pane > Click 'Browse DBFS' to list DBFS file as shown on the second left pane from the screenshot1 > Click the folder  'NORMALIZED\_ATTRIBUTE\_OUTPUTS' to show all the files in folder > locate the file you want to download and click 'arrow' to show the possible actions > Click 'Copy path'  to show the 'Select a path to copy' screen like Select a path to copy> Click 'Copy' under 'File API Format' > The 'copy' will change to 'Copied' like screenshot3 > Click  close to get copy path

### Attribute Match Process

Fuzzy matching algorithms calculate a similarity score between two strings, allowing for partial matches. It helps improve the accuracy and efficiency of matching tasks.

RapidFuzz is a Python library that provides fast and efficient fuzzy string matching algorithms.

The Fuzz module in RapidFuzz provides various functions and algorithms for fuzzy string matching and comparison. These functions can be used to perform fuzzy matching, similarity calculations, and other fuzzy string operations in RapidFuzz. Here are some notable functions provided by the Fuzz module in RapidFuzz:

token\_set\_ratio: Calculates the similarity between two strings by considering the unique tokens present in both strings.

partial\_ratio: Calculates the similarity between two strings by considering partial matches and short strings.

process: Provides functions like extract and extractOne for finding the best matches between a query string and a list of choices.

It offers all the unique ways in which you can compare two pieces of strings.

The process.extract function in RapidFuzz returns a list of tuples containing the matched choices along with their similarity scores, sorted in descending order. It is used to find the best matches between a query string and a list of choices, based on string similarity.

I apologize for the confusion in my previous response. It seems I provided information about the fuzzywuzzy library instead of RapidFuzz.

In RapidFuzz, the process.extract function is also used for fuzzy string matching, but with a slightly different syntax and functionality. Here is the correct information for process.extract in RapidFuzz:  
  
The process.extract function in RapidFuzz is used to find the best matches between a query string and a list of choices, based on string similarity.  
  
Here is the syntax of the process.extract function in RapidFuzz:

rapidfuzz.process.extract(query, choices, scorer=rapidfuzz.fuzz.WRatio, limit=5)

* query: The string for which you want to find the closest matches.
* choices: A list of strings that you want to compare the query string against.
* scorer: An optional argument that defines the scoring algorithm to be used for matching. The default scorer is rapidfuzz.fuzz.WRatio, which calculates the similarity using weighted ratio.
* limit: An optional argument that specifies the maximum number of matches to be returned. The default value is 5.

score\_cutoff is used to set a minimum similarity for the string matching. Elements below this similarity will not be added to the results. Providing this parameter helps to improve the performance, since it allows rapidfuzz to use faster algorithms to compare strings  
  
The process.extract function in RapidFuzz returns a list of tuples containing the matched choices along with their similarity scores, sorted in descending order.

**Ratio:**It calculates the normalized distance.

**Partial Ratio:**It finds the ratio similarity measure between the shorter string and every substring of length m of the longer string, and returns the maximum of those similarity measures. Basically, it searches for the optimal alignment of the shorter string in the longer string and returns the fuzz.ratio for this alignment.

**Token Set Ratio:** Compares the words in the strings based on unique and common words between them. It takes a set of all the tokens in the string and then compares it. If String A has all the elements of String B, the ratio will be 100. But it’s not true the other way around.

The process.extract function in RapidFuzz is used for fuzzy string matching,

Key features of RapidFuzz include

token\_set\_ratio

fuzz.partial\_ratio

1. Create a dataset 'item' which contains 'DESCRIPTION' from the Input file
2. Function attribute\_match

Invoke rapidfuzz.process.extract twice to match the value of attribute.

The first call will pass 'DESCRIPTION'   from 'ITEM' dataset, attb\_list, and attb\_subs from the config file to the function attribute\_match to get

1. 'Normalized' attribute.

used “fuzz.token\_set\_ratio” and cut\_off value to match the value of attribute.

The second call used “fuzz.token\_partial\_ratio” to match list from the first call to match the value of attribute.

The fuzz\_score will compare with the cut-off

attribute

 and Invoke rapidfuzz.process.extract twice to match attribute.

[PIM\_COLOR\_EXTRACTION\_111723.ipynb.txt](http://localhost:8888/edit/Kroger/Code/PIM_COLOR_EXTRACTION_111723.ipynb.txt)

1. Create a dataset 'item' which contains 'DESCRIPTION' from the Input file
2. Function attribute\_match

Invoke rapidfuzz.process.extract twice to match the value of attribute.

The first match will pass 'DESCRIPTION'   from 'ITEM' dataset, attb\_list, and attb\_subs from the config file, limit= 7, score\_cutoff= 90, scorer = fuzz.token\_set\_ratio to rapidfuzz.process.extract and return the match list.

The re-match will pass 'DESCRIPTION'   from 'ITEM' dataset, match\_list from the first match, scorer = fuzz.partial\_ratio, limit = 5 to rapidfuzz.process.extract and return the final-match list.

Reference for

to the function attribute\_match to get

1. 'Normalized' attribute.

used “fuzz.token\_set\_ratio” and cut\_off value to match the value of attribute.

The second call used “fuzz.token\_partial\_ratio” to match list from the first call to match the value of attribute.

The fuzz\_score will compare with the cut-off

attribute

 and Invoke rapidfuzz.process.extract twice to match attribute.

PIM\_COLOR\_EXTRACTION\_rugs\_120523.ipynb

pim\_dpt['EXTRACTED\_COLOR']  =  \

                pim\_dpt.MATCHING\_DESCRIPTION.str.upper().replace(replacements, regex = True).apply(lambda x:color\_match(x, color\_list\_org= color\_list))# + [i.replace("

[Yesterday 12:02 PM] Yue, Sophia (NonEmp)

Hi MX Core Data (Public),  may I have the link or document to apply for the access of  'mx-core-data-stage-dbxws', please?

[Yesterday 12:15 PM] Patel, Parav Kumar (NonEmp)

Yue, Sophia (NonEmp) here is the onboarding document. You need to raise APT access for specific domain. Please check this document: [MX Data Platform - MX Core - Confluence (kroger.com)](https://confluence.kroger.com/confluence/display/MC/MX+Data+Platform)

Since the user may be new to Databricks/Python,  we want to explain:

1) How to run a code cell by clicking the arrow( shown in screenshots)

2) Remind that the cell needs to be run first before they can input something ( selecting attribute, dept, or uploading the file)

Hi Patel,

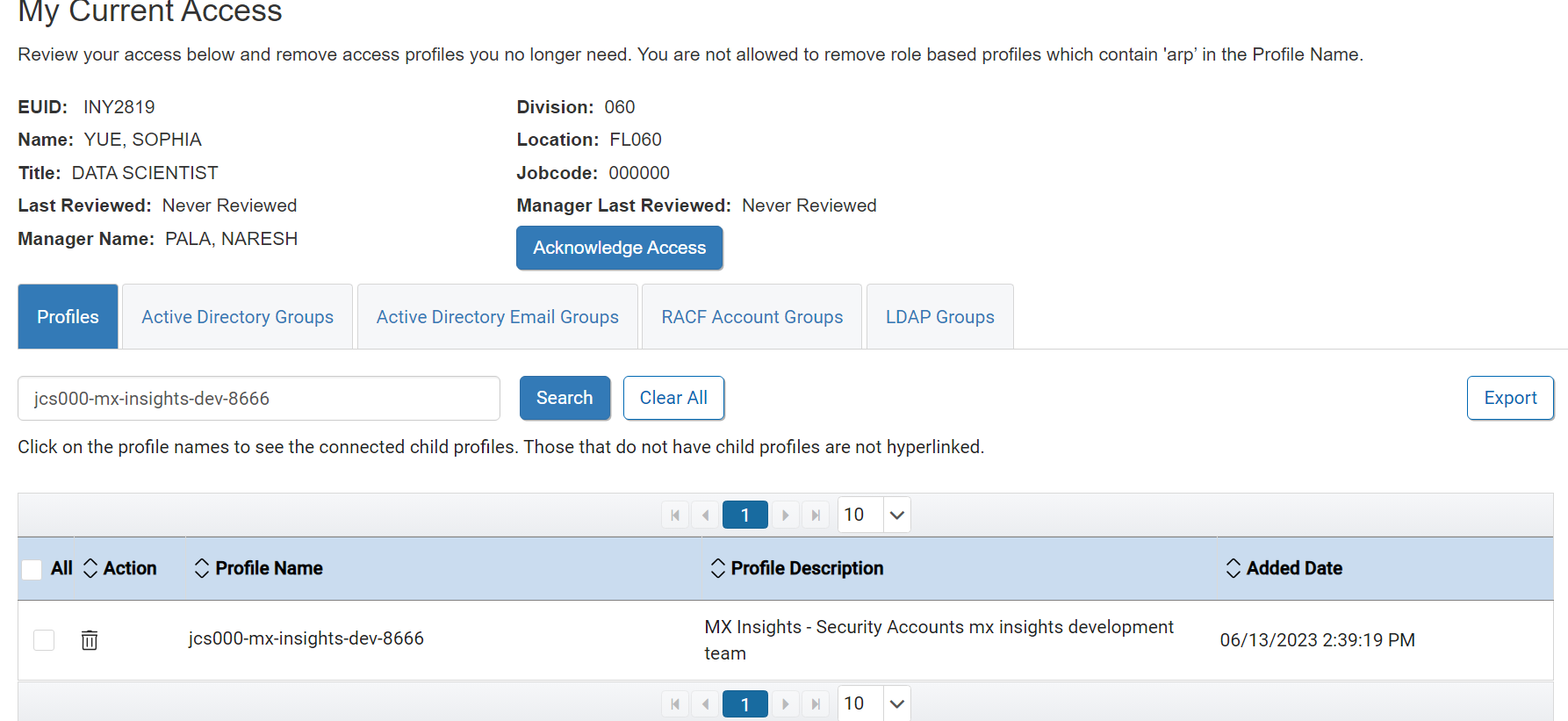
Thank you for respondibng to my message. I need to include how to get access to Databricks in my Confluence page for the new use who will have the same domain name as I have.

From AIM > [Access Management](https://iam.kroger.com/access-management/my-current-access) > [My Current Access](https://iam.kroger.com/access-management/my-current-access),  I have access to jcs000-mx-insights-dev-8666  and I didn’t see jcs000-mx-insights-dev-8666**-domain name.**

The document [MX Data Platform - MX Core - Confluence (kroger.com)](https://confluence.kroger.com/confluence/display/MC/MX+Data+Platform) suggest raise APT for  jcs000-mx-insights-dev-8666**-domain name to access** MX Dataplatform.

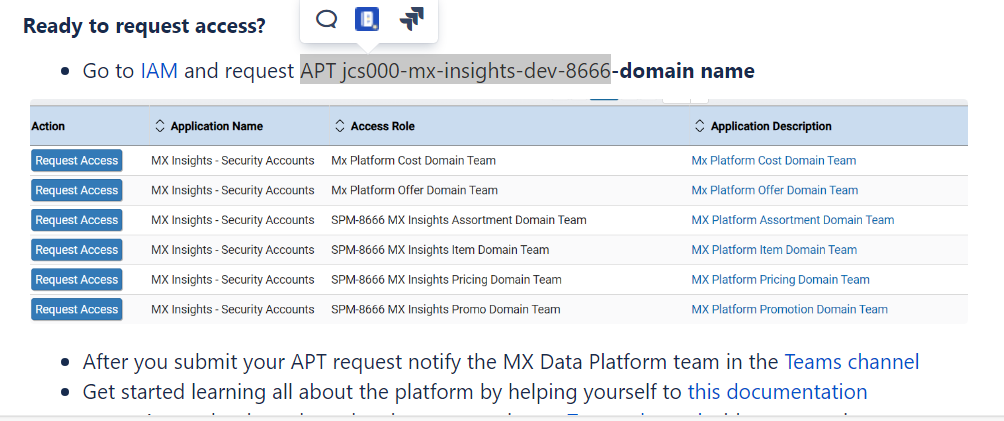
T**o access** MX Dataplatform, does the new user need to raise the APT for jcs000-mx-insights-dev-8666 or jcs000-mx-insights-dev-8666**-domain name?**

**I have access to ‘mx-core-data-stage-dbxws'. How do I find my domain name?**



[MX Data Platform - MX Core - Confluence (kroger.com)](https://confluence.kroger.com/confluence/display/MC/MX+Data+Platform)

Yue: I have access jcs000-mx-insights-dev-8666. When raise APT, it might nte required domain name.



Per Patel from ‘MX Core Data (Public)’ team, the new profile to access MX Dataplatform is jcs000-mx-insights-dev-8666**-domain name** instead of jcs000-mx-insights-dev-8666 which is old APT for MX Dataplatform.  Per Patel, the new profile is domain-specific and we can get the **domain name from you.**

Mallik's notebook: [Attribute\_Extraction\_Prod - Databricks (azuredatabricks.net)](https://adb-4812933386228410.10.azuredatabricks.net/?o=4812933386228410#notebook/1949532348594449/command/1949532348594457)

### **Attribute / Product Category schema as of 1/18/2024**

|  |  |
| --- | --- |
| **Product Category** | **Attribute** |
| Floral | Color |
| Home Decor | Color |
| Kitchen & Dining | Color |
| Beverages | Flavor |
| Candy | Flavor |
| Ice Cream | Flavor |
| Pantry | Flavor |
| Beauty | Scent |
| Cleaning & Household | Scent |
| Personal Care | Scent |

How to run the code

The code is saved in Azure Databricks notebooks.

Notebooks in Azure Databricks

Notebooks in Azure Databricks are web-based interfaces and provide an interactive environment where you can write code, execute it, and view the results in real-time.

Our script ‘automation of normalized attribute extraction for new items’ is a Databricks notebook written in Python with two types of cell:

Markdown cells are for document purpose.

Code cells with/without cell title are codes. After navigating the script. Run the code cells in sequence.

], "choices": [ { "finish\_reason": "stop", "index": 0, "message": { "role": "assistant", "content": "{'Color': 'YELLOW/ORANGE', 'Confidence': 90}" },

**From:** Agarwal, Vivek (NonEmp)  
**Sent:** Tuesday, January 9, 2024 12:56 PM  
**To:** Chakraborty, Shiladitya (NonEmp) <[shiladitya.chakraborty@kroger.com](mailto:shiladitya.chakraborty@kroger.com)>; V Ray, Shashank (NonEmp) <[shashank.vray@kroger.com](mailto:shashank.vray@kroger.com)>; Dua, Prateek (NonEmp) <[prateek.dua@kroger.com](mailto:prateek.dua@kroger.com)>; Yue, Sophia (NonEmp) <[sophia.yue@kroger.com](mailto:sophia.yue@kroger.com)>; Sarkar, Suman (NonEmp) <[suman.sarkar@kroger.com](mailto:suman.sarkar@kroger.com)>; M N, Mallik (NonEmp) <[mallik.mn@kroger.com](mailto:mallik.mn@kroger.com)>; Nair, Deevraj (NonEmp) <[deevraj.nair@kroger.com](mailto:deevraj.nair@kroger.com)>; Aketi, Prasanth (NonEmp) <[prasanth.aketi@kroger.com](mailto:prasanth.aketi@kroger.com)>; [vivek.agarwal06@infosys.com](mailto:vivek.agarwal06@infosys.com) <[vivek.agarwal06@infosys.com](mailto:vivek.agarwal06@infosys.com)>  
**Cc:** Sundarrajan, Vyas <[vyas.sundarrajan@kroger.com](mailto:vyas.sundarrajan@kroger.com)>; Rajkumar, Mini <[mini.rajkumar@kroger.com](mailto:mini.rajkumar@kroger.com)>; Friemel, Paul A <[paul.friemel@kroger.com](mailto:paul.friemel@kroger.com)>; Shashank Ray <[shashank.ray@infosys.com](mailto:shashank.ray@infosys.com)>  
**Subject:** Sprint Retrospective MX: Finding Nemo  
**When:** Tuesday, January 23, 2024 7:30 PM-8:00 PM.

### JSON file basics

* A JSON file, also known as JavaScript Object Notation, is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.
* The extension of a JSON file is json.
* JSON objects
  + JSON objects are a fundamental component of JSON data structure.
  + They represent a collection of key-value pairs enclosed in curly braces {}.
  + Each key is a string, followed by a colon (:), and then a value. Multiple key-value pairs are separated by commas (,).
  + The values can be strings, numbers, booleans, null, other objects, or arrays. Arrays represent an ordered list of values, which can be any valid JSON data type.
  + The keys within an object must be unique.
  + Example of JSON objects

{"CFIC\_DPT": "Floral",  
 "Attribute": "COLOR",   
 "Normalized\_list": ["RED", "ROSE", "WHITE", .....,"PINK", "RED"],   
 "Substitutions": {"ASSORTED": "MULTICOLOR", "ASSORTED COLORS": "MULTICOLOR", ....,"DARK PINK": "PINK", "DARK RED": "RED"}}

A JSON file is a collection of key-value pairs enclosed in curly braces {}. Each key is a string, followed by a colon (:), and then a value. Multiple key-value pairs are separated by commas (,). The keys within an object must be unique.

 ptext editorroduct category  from Paul ????

# User guide of [**Automation of normalized attribute extraction for new items**](https://confluence.kroger.com/confluence/display/EIS/Automation+of+normalized+attribute+extraction+for+new+items)

[run book of normalized attribute extraction for new items - EIX - Enterprise Item Experience - Confluence (kroger.com)](https://confluence.kroger.com/confluence/pages/createpage.action?spaceKey=EIS&atl_token=e92cdd0971d7e38f43135d0d41702e2dae4ede35&fromPageId=412108506)

SELECT  item.GTIN\_NO AS GTIN, atb.VND\_ECOM\_DSC AS DESCRIPTION, cfic.ITM\_CFIC\_FYT\_DPT\_NAM AS

DEPARTMENT, cfic.ITM\_CFIC\_FYT\_COM\_NAM AS COMMODITY, item.GTIN\_ITM\_SBM\_DT AS ITEM\_SUBMISSION\_DATE

FROM curidq.pimmart.ITEM\_VIEW as item

LEFT JOIN curidq.pimmart.ITM\_ATB\_LNG as atb ON item.ITM\_ID =   atb.ITM\_ID

LEFT JOIN curidq.pimmart.ITM\_CFIC\_FYT as cfic ON item.ITM\_ID =   cfic.ITM\_ID

WHERE ((cfic.ITM\_CFIC\_FYT\_DPT\_NAM IN ('Floral', 'Beverages', 'Beauty', 'Candy','Pantry','Home Décor','Cleaning and Household','Personal Care','Kitchen & Dining')  OR cfic.ITM\_CFIC\_FYT\_COM\_NAM = 'Ice Cream')

AND item.GTIN\_ITM\_SBM\_DT  >= ' yyyy-mm-dd');

If the last run date is 2023-12-01, the yyyy-mm-dd should be 2023-12-01 [Edit - User guide to create an output file of normalized attribute extraction for new items - EIX - Enterprise Item Experience - Confluence (kroger.com)](https://confluence.kroger.com/confluence/pages/editpage.action?pageId=434782439)

save SQL server query result to a csv file

To save the result of an SQL Server query to a CSV file, you can use the following steps:

1. Execute the query: Run your SQL Server query using a SQL Server management tool such as SQL Server Management Studio (SSMS) or a programming language's database connector.
2. Export the query result: Once you have the query result displayed, right-click on the result grid and select the "Save Results As..." or "Export Data..." option. Alternatively, you can use the "Query" menu and choose "Results to File" or a similar option depending on the tool you are using.