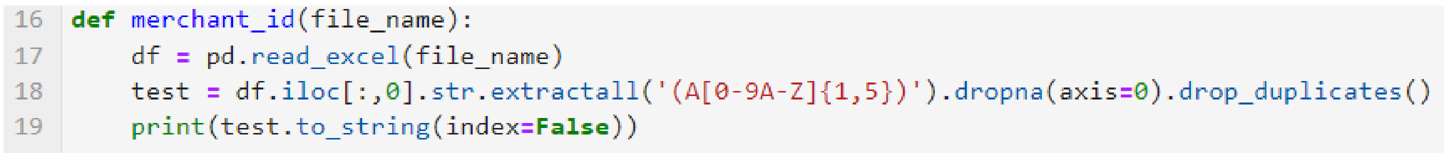
Python script – key data extraction guide

1. *Jupyter notebook IDE and code structure*

Jypyter lab Interactive Development Environment (IDE) was selected since it is an easy-to-use interface with minimal page obstructions and also upload excel files directly using the file browser option. This was a personal preference, but the code can be deployed in any other IDE such as Spyder, lab etc. to get the same results.

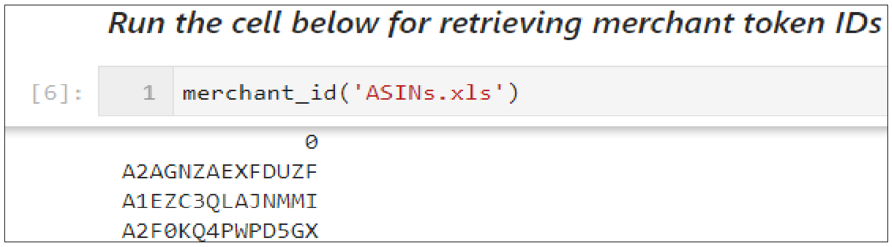
The code is structured for ease in readability, running and obtaining results. Functions are used to clear clutter of repeating codes and clearly naming the process for which data is to be extracted for example, **def asins_xb ()** only extracts ASINs that begin with X or B. Whereas **def merchant_id ()** only extracts merchant IDs that begin with A and have a range of minimum 10 to 17 characters of either number (0 -> 9) **OR** an upper case alphabet (A->Z)

2. *Quick examples*



```
16  def merchant_id(file_name):
17      df = pd.read_excel(file_name)
18      test = df.iloc[:,0].str.extractall('(A[0-9A-Z]{1,5})').dropna(axis=0).drop_duplicates()
19      print(test.to_string(index=False))
```

- Line 16: Create a function named “merchant_id” that accepts input called “file_name” (excel file which must be uploaded by associate). In this example, the uploaded file name in folder was “ASINs.xls” (see screenshot below) when calling the function.



Run the cell below for retrieving merchant token IDs

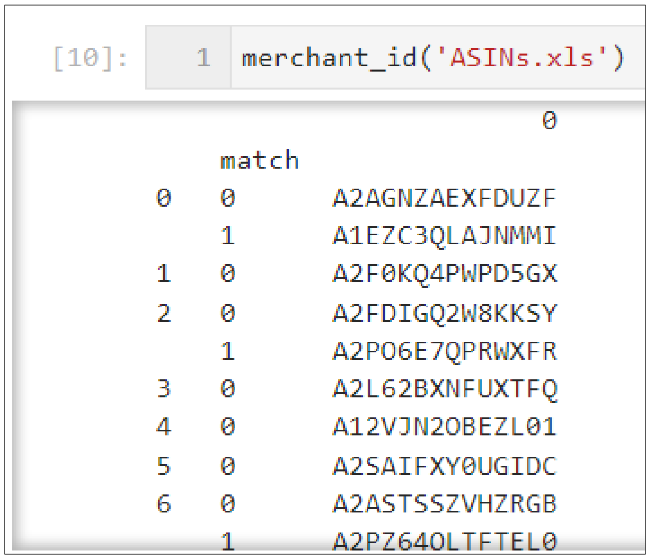
```
[6]:  1  merchant_id('ASINs.xls')
```

```
             0
A2AGNZAEXFDUZF
A1EZC3QLAJNMMI
A2F0KQ4PWPD5GX
```

- Line 17: Use pandas to read the excel file into a dataframe for the string extraction process
- Line 18: use the iloc method to choose all rows but only the first column from which strings are extracted using a defined condition logic. This is further accompanied by dropping duplicates and removing any missing values only from rows i.e. (axis = 0) argument
- Line 19: Finally, print the results in the output cell to visually inspect the extracted merchant IDs. Optionally, user can use the test.to_clipboard(index = False) to copy all output data directly onto their windows clipboard (accessed by using windows key + v) that can save time instead of copying the entire cell manually

3. *Extraction accuracy & data validation check*

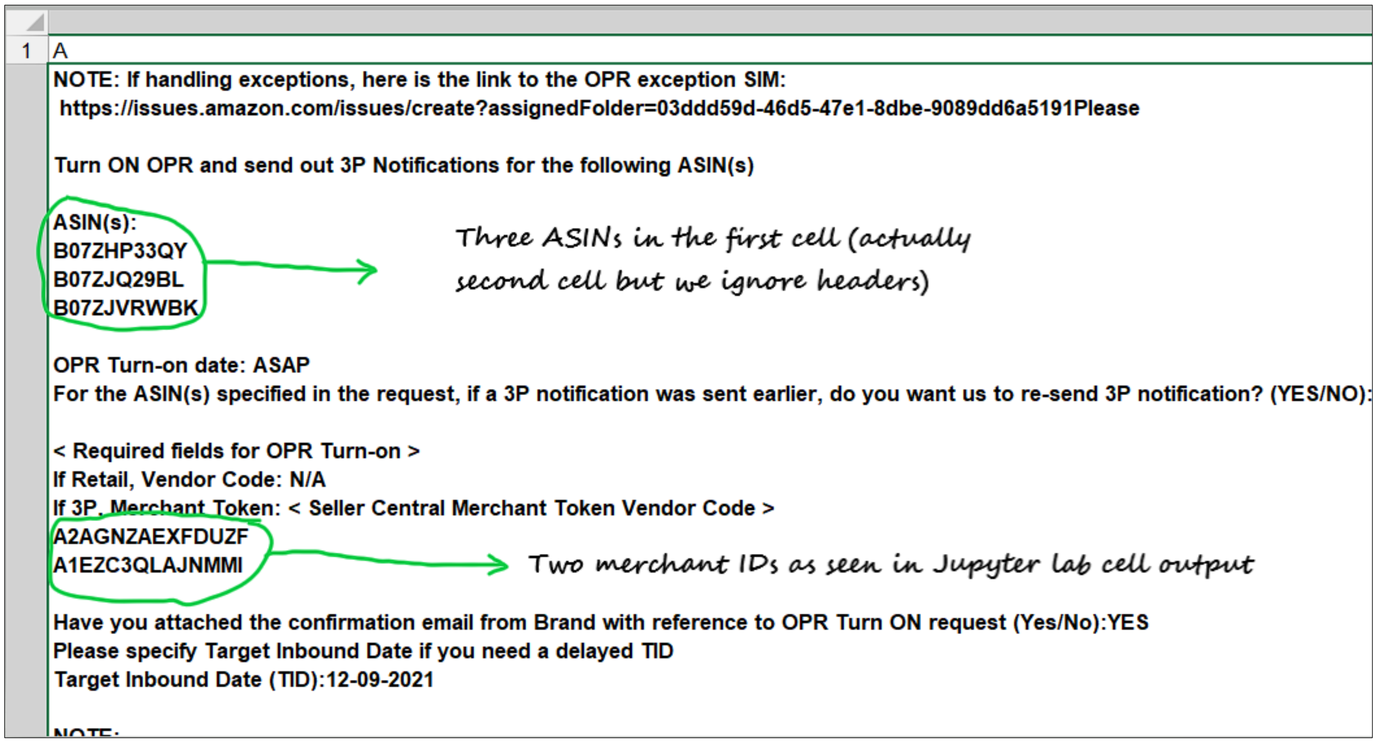
To ensure data integrity post script execution, it was done by deep-diving to crosscheck top 10 excel sheet cells where ASIN/merchant ID data was present. To further simplify the cross reference, the argument (index = False) was removed from the print statement (line 19) and script was run. The output is given as follows,



```
[10]:  1  merchant_id('ASINs.xls')

                       0
          match
   0      0       A2AGNZAEXFDUZF
          1       A1EZC3QLAJNMMI
   1      0       A2F0KQ4PWPD5GX
   2      0       A2FDIGQ2W8KKSY
          1       A2PO6E7QPRWXFR
   3      0       A2L62BXNFUXTFQ
   4      0       A12VJN2OBEZL01
   5      0       A2SAIFXY0UGIDC
   6      0       A2ASTSSZVHZRGB
          1       A2PZ64OLTFTEL0
```

We can see that the structure is indexed using excel cell format i.e. the match column shows number of merchant IDs extracted and the preceding column shows cell position in excel. Since Python uses 0 as a first count, this means that the first cell in excel sheet (excluding header) has two merchant IDs which are **A2AGNZAEXFDUZF** and **A1EZC3QLAJNMMI** respectively. This can be crosschecked directly from the excel file as shown below:



1 A

NOTE: If handling exceptions, here is the link to the OPR exception SIM:
https://issues.amazon.com/issues/create?assignedFolder=03ddd59d-46d5-47e1-8dbe-9089dd6a5191Please

Turn ON OPR and send out 3P Notifications for the following ASIN(s)

ASIN(s):
B07ZHP33QY
B07ZJQ29BL
B07ZJVRWBK

Three ASINs in the first cell (actually second cell but we ignore headers)

OPR Turn-on date: ASAP
For the ASIN(s) specified in the request, if a 3P notification was sent earlier, do you want us to re-send 3P notification? (YES/NO):

< Required fields for OPR Turn-on >
If Retail, Vendor Code: N/A
If 3P, Merchant Token: < Seller Central Merchant Token Vendor Code >
A2AGNZAEXFDUZF
A1EZC3QLAJNMMI

Have you attached the confirmation email from Brand with reference to OPR Turn ON request (Yes/No):YES
Please specify Target Inbound Date if you need a delayed TID
Target Inbound Date (TID):12-09-2021

NOTE:

The very same can be verified for remaining cells. So far, there is not a single data chunk (ASINs, merchant IDs, 10-digit merchant ID) that was missed.

4. *Search logic in-depth look*

The key for this script to work as intended was the design and thorough testing of the search logic. This will be explored below for all 3 unique data chunks:

- For the ASINs

str.extractall(**'([XB]0.......[0-9a-zA-Z])'**)

Alternatively, **str.extractall**(**'([XB]0.{7}[0-9a-zA-Z])'**)

Standard protocol for ASINs forge them to either begin with an *X* **OR** *B* followed by an immediate *0* and ends with **either** a *0 to 9* **OR** upper/lower case alphabet *a to z* which is overall 10 characters in length. This condition is met using logic shown above. Breakdown of individual components as shown below,

[XB] : Include either an X or a B

0 : Only include 0 as the second character

. : This dot means choose any character which includes special and alphanumeric texts

[0-9a-zA-Z] : **either** a *0 to 9* **OR** upper/lower case alphabet *a to z*

{7} : A repetition which means **repeat** this range for another 6 characters i.e. **7** characters in total

- For the 10-digit ASIN code

str.extractall(**'([0-9]{10})''**)

Standard protocol for these ASINs forge them to either begin with range of *0 to 9* having overall 10 characters in length. This condition is met using logic shown above. Breakdown of individual components as shown below,

[0-9] : Include range of digits **0 to 9**

{10} : A repetition which means **repeat** this range for another 9 characters i.e. **10** characters in total

- For the Merchant IDs

str.extractall(**'(A[0-9A-Z]{10,17})'**)

Standard protocol for merchant IDs forges them to only begin with an *A* followed by *0 to 9 digit* ***OR*** upper-case alphabet *A to Z* with repetition of overall minimum 10 characters OR 16 characters in length. This condition is met using logic shown above. Breakdown of individual components as shown below,

A : Include only an A

[0-9A-Z] : **either** a *0 to 9* **OR** upper case alphabet *A to Z*

{10,17} : Repeat this range of **[0-9A-Z]** minimum 10 characters to maximum 17 characters i.e. overall 11 to 18 characters

Merchant IDs like these; A2RWE9T9AXOGK1, A1K4IFKV616H, A2Q9GXD6UCW405, A14SMJT84YF6S9, A4ZGBZ1G1A6U9, AP4R4A1EYFYA are all considered when using the extraction technique. Even though the character lengths are varying, none are omitted ensuring high accuracy.

5. *Potential uses and further implications*

The code used was Python 3 (kernel) which means this script can be slightly modified to be adaptively used by AWS Lambda as a trigger to extract key data chunks which can be modified as desired by the user. This means the process is scalable as the code can be used for any AWS resource or data in warehouses with proper in-line policies and defined event types (using IAM policies).