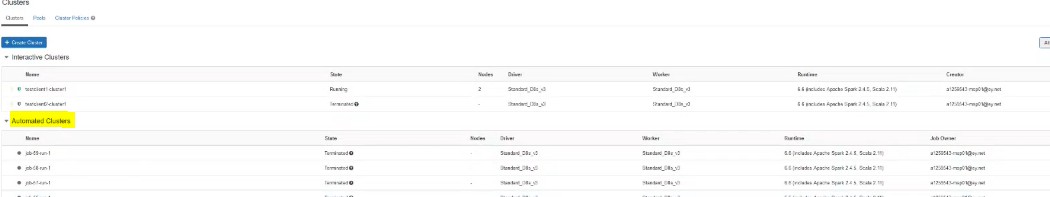
STARTING THE INGESTION JOB/CD INGESTION PIPELINE

Anytime if the request is kick started , It basically sets up an automated cluster

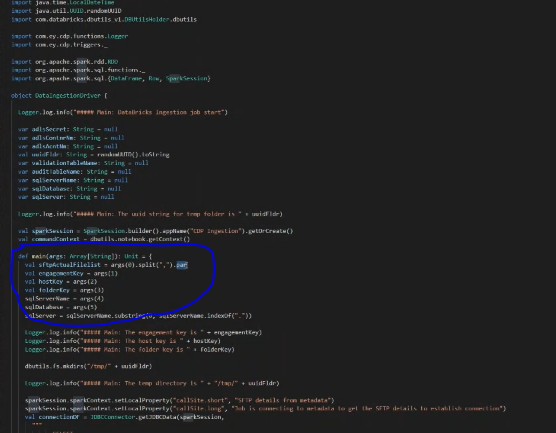


If any of the cluster is opened up and in the job run the below details can be found

* This automated cluster is fired from the CD orchestrator Azure functions
* Jar request
* Arguments passed-filename,engagement details,table details
* SQL connections where the metadata is stored

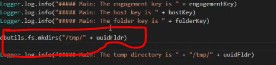
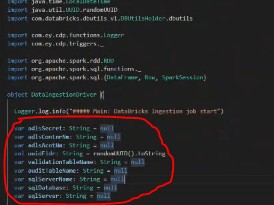
So once it starts running the settings are pulled here in to these particular values so you have your

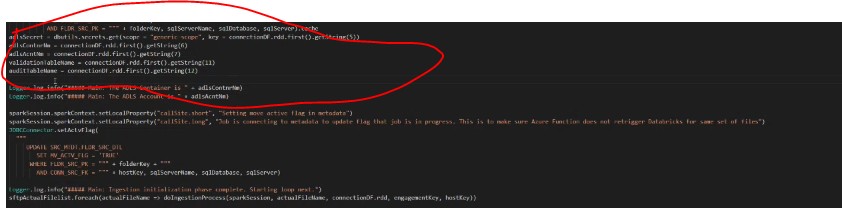
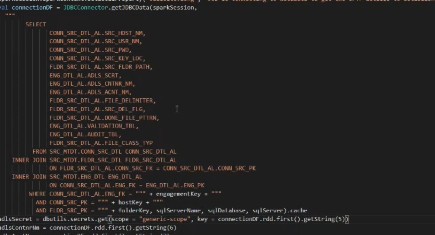
SFTP Filelist ,your engagement key ,host key. The below piece of code highlighted exhibits parallelism and it explains that the number of files which is ready for processing can run simultaneously



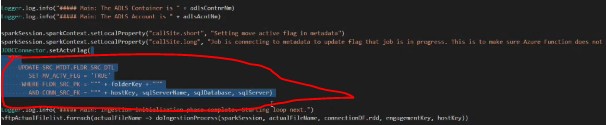


In the below code we had these 5 hashes to distinguish it from the bunch of logs from other databricks output like creating a uid string which is the place where temporary folders will be created and getting the sql server and whatever operations will do will be inside this temp directory .

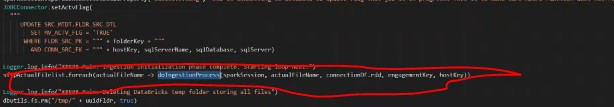


The below code represents the first query we are firing and this is not a database query .we are fetching details like FTP host,username,password,ADLS secret container name, file delimiters tat need tobe implemented for a file and allthese info and put in to the connection DF

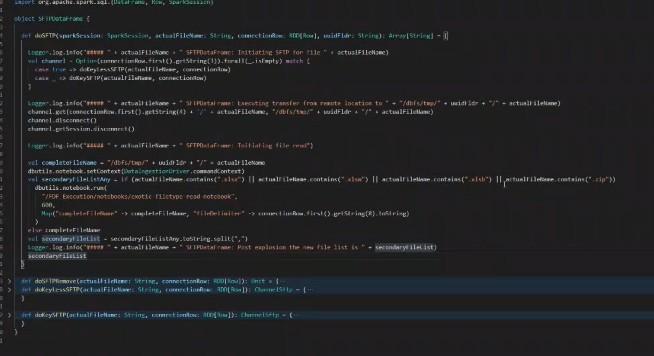
Then the flag value is set to true then next time when the azure function runs it shold not kickstart another databricks job for the same set of files



In the below code it takes foreach execution it takes everyfile name in list and does this ingestion process simultaneously and this where it exhibits multithreading feature



The below piece of code helps in converting xls files zip files in to csv and text formats



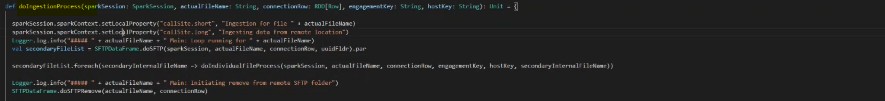
In the below function named as **dataingestionprocess** the following opeaartuons are performed

* Individual file processing
* Vaildations -Filelevel& Rowlevel
* Writing the file in to the target and writing the audit table
* Deleting the actual filename



In the below function named as **doindividualfileprocess** the following operations are performed

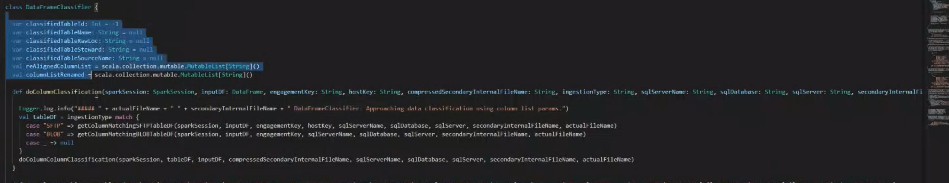
* CSV read
* Compressed Filename – removes special characters



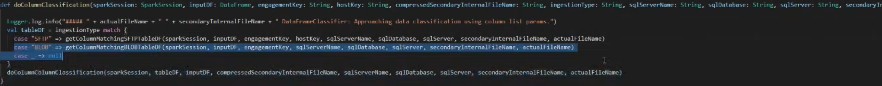
The below piece of code checks if a particular column value is null we can go with the assumption that everything needs to be dropped after it



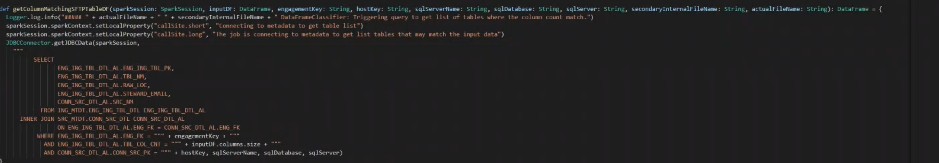
The client just drop a file in to the SFTP location . The below code act as entire dataframe classifier and it includes two ways , one is to do column matching where it takes the column values and matches with the column names in the metadata and it writes the data in to the table and the other is based on the filename matching where whatever parameters we have passed if it matches the like condition it is going to write the data. Dataframe classifier has couple of values and this is the only class in the entire code . We need to create a function for it



In the below we are doing column classification and the first step is trying to identify whats the pattern is it SFTP or BLOB

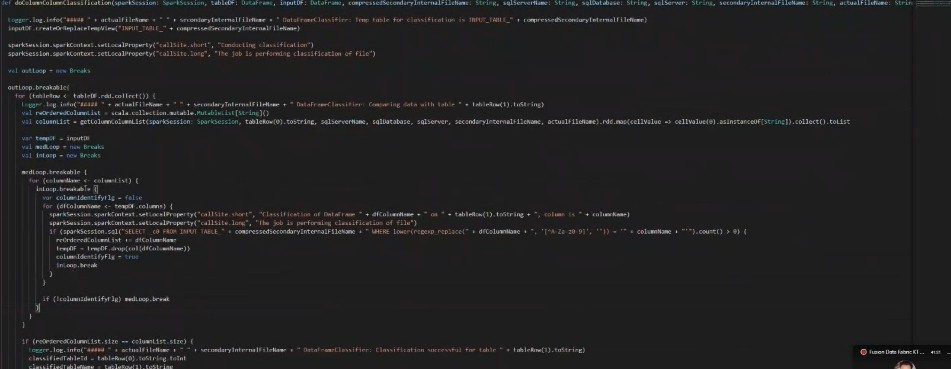


If it is SFTP in this case for whatever that engagement key it is getting only that table list where the column name matches table column count matches



This function **docolumncolumnclassification** does the following

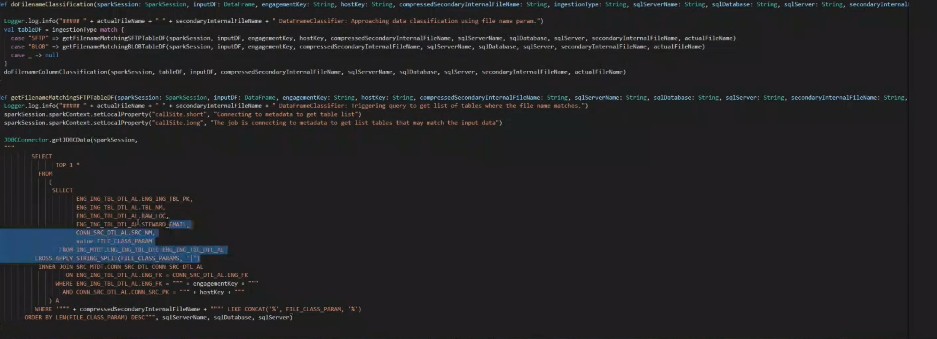
Compressed Secondary Internal filename value to almost any temporary table we create for removing any special characters like dots and hashes and this is where the classification is happening . This code helps to match the column in all cases and if your two lists column list and the ordered list are of the same size then it is good to proceed



So taking this one as column levelmatching



If it is filename classification where filename parameters are going to match so it Is like taking the compressed file name and doing the like operation

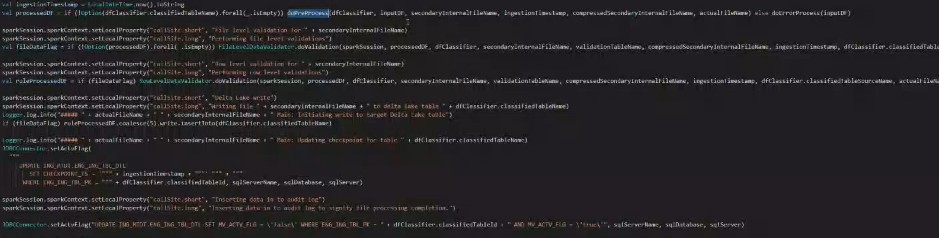


Now we have the Compression filename ,read of the file and the classification piece too

The below codeis the use case when u have multiple files writing to the same table

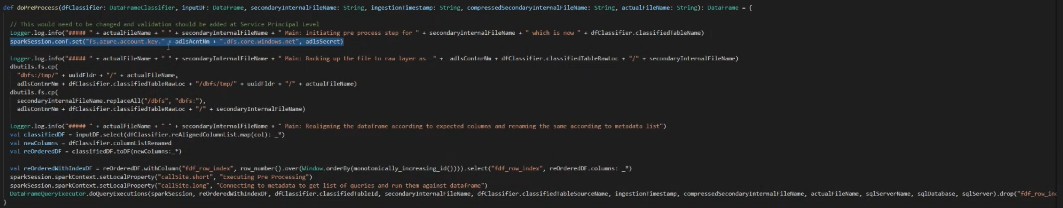


After this process scrubbing of data happens with the below code . We are running a bunch of sqls on a input dataframe and we are creating a output dataframe



There is function named as **datapreprocess**

Setting up your ADL secret ,which gives you access to that particular adls. Here the back up files are created (10:50-11:10)



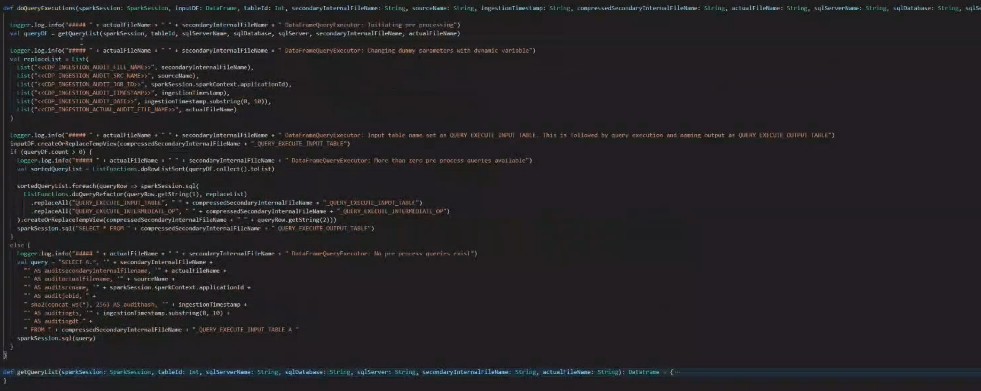
Then we have to create a reorder dataframe for the data which we are storing in to the target table and this make sure that column position is not affected



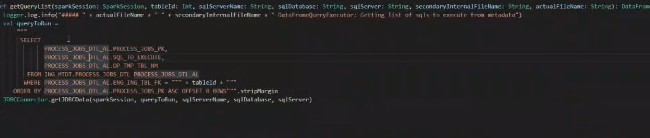
We are adding the below code for adding the montonically increasingid called as FDFrowindex and aadding this to dataframe



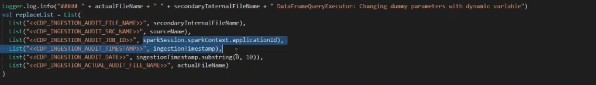
Then we come to the function **doqueryexecution.** It gets the query list to be executed in sequential order on top of that particular dataframe



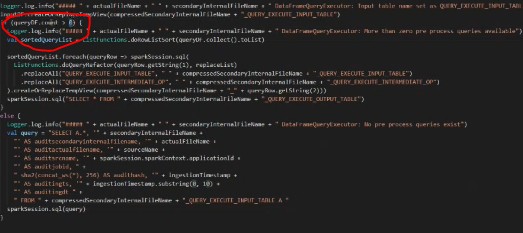
And that query list is metadata driven



Parametrs passed like Internalfile name,conection source details etc



The below piece of code shows that if querydfcount>0 that it is possible that for a ceratin table there will be no query logic to execute in that case it is written directly to the target table

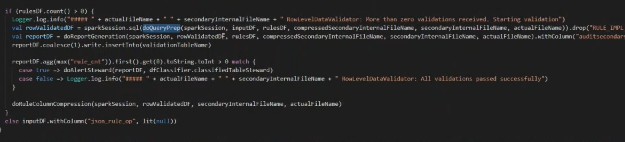


If we have our processed DF ready there is actually an if condition before which says if the table got successfully classified then its going to do the preprocesss but if it is not then there will be a mail notification on it

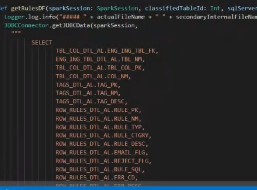


Then if we get our process dataframe successfully we need to do two things one is file level validation and the other is the rowlevel validations. After completing the requested validations it will write to the client specific validationtable.

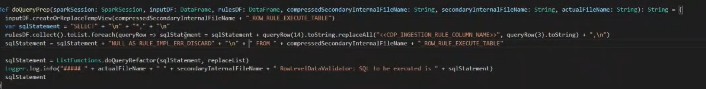
If there are some rules that needs to be checked for a table and query needs to be prepared for running those rules



There is a function called **getrulesdf**

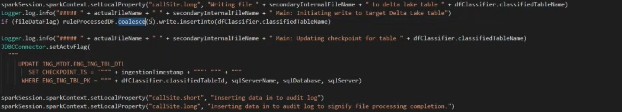


All the rules are executed in one query with a function named as **doqueryprep**



Once we get the report df we write both the filelevel validations and rowlevel vaildations in to the validation table

Writing the ruleprocessed df in to the target table i.e whetever the table classified and coalesce (5) is used for writing the table and it updates the chechpoint write the audit details



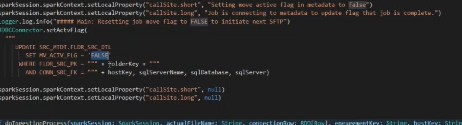
So after this we finish up the entire file received which is mentioned in the line number 98 once that’s done



we are removing the temporary databricksDBFS path



And we are setting the mv\_actv\_flag as false and now azure functions knows that now I can search for any more additional files on the SFTP site and this



Concludes the entire ingestion process