

# PYSPARK NOTEBOOK

C O N T E N T S

Set up a job (inception)	1
Read from table / show 3 records	2
Check data types	3
Count The Number of rows in a dynamic dataframe	4
Select specific fields from a dynamic dataframe	5
Drop fields from a dynamic dataframe	6
Mapping an array to allow a column rename to take place	7
Filter a dynamic frame for customers who have the last name	8
Join 2 dynamic frames using an equality join	9
Write Down Data from a Dynamic Frame using Glue Data Catalog	10
Convert from Dynamic Frame To Spark DataFrame	11
Selecting columns in a spark dataframe	12
Add Columns In A Spark Dataframe - creating a new column with a literal string	13
Using concat to concatenate two columns	14
Dropping columns	15
Renaming columns	16
GroupBy and Aggregate functions	17
Filtering columns and where clauses	18
Joins - read the orders table and convert it to a spark dataframe	19
Inner joins for spark dataframes	20
Inner joins for field members with a specified surname	21
Left joins	22
Writing data down using the glue data catalog	23
Writing data to an S3 location	24
Write data to a glue table using glue catalog metadata	25

1.Set up a job (inception)	<pre># inception  import sys from awsglue.transforms import * from awsglue.utils import getResolvedOptions from pyspark.context import SparkContext from awsglue.context import GlueContext from awsglue.job import Job  sc = SparkContext() glueContext = GlueContext(sc) spark = glueContext.spark_session job = job(glueContext)</pre>
2. Read from table / show 3 records	<pre># read from the customers table in the glue data catalog using a dynamic frame  dynamicFrameCustomers = glueContext.create_dynamic_frame.from_catalog(     database = "pyspark_tutorial_db",     table_name = "customers" )  # show the top 10 rows from the dynamic dataframe  dynamicFrameCustomers.show(3)</pre>
3. Check data types	<pre># check types in dynamic frame  dynamicFrameCustomers.printSchema()</pre>
4. Count The Number of Rows in a dynamic dataframe	<pre># count The Number of Rows in a dyanamic dataframe  dynamicFrameCustomers.count()</pre>
5. Select specific fields from a dynamic dataframe	<pre># selecting certain fields from a dynamic dataframe dyfCustomerSelectFields = dynamicFrameCustomers.select_fields(["customerid", "fullname"])  # show top 10 dyfCustomerSelectFields.show(10)</pre>

6. Drop fields from a dynamic dataframe	
	<pre> # drop Fields of Dynamic Frame dyfCustomerDropFields = dynamicFrameCustomers.drop_fields(["firstname","lastname"])  # show Top 10 rows of dyfCustomerDropFields Dynamic Frame dyfCustomerDropFields.show(10) </pre>
7. Mapping an array to allow a column rename to take place	
	<pre> # mapping array for column rename fullname -&gt; name mapping=[("customerid", "long", "customerid","long"),("fullname", "string", "name", "string")]  # apply the mapping to rename fullname -&gt; name dfyMapping = ApplyMapping.apply(     frame = dyfCustomerDropFields,     mappings = mapping,     transformation_ctx = "applymapping1" )  # show the new dynamic frame with name column dfyMapping.show(10) </pre>
8. Filter a dynamic frame for customers who have the last name	
	<pre> # filter dynamicFrameCustomers for customers who have the last name Adams dyffilter= Filter.apply(frame = dynamicFrameCustomers,                         f = lambda x: x["lastname"] in "Adams"                         )  # show the top 10 customers from the filtered Dynamic frame dyffilter.show(10) </pre>
9. Join 2 dynamic frames using an equality join	
	<pre> # read from the customers table in the glue data catalog using a dynamic frame dynamicFrameOrders = glueContext.create_dynamic_frame.from_catalog(     database = "pyspark_tutorial_db",     table_name = "orders" )  # show top 10 rows of orders table dynamicFrameOrders.show(10)  # join customers and orders dynamic frame dfyjoin = dynamicFrameCustomers.join(["customerid"],["customerid"],dynamicFrameOrders)  # show top 10 rows for the joined dynamic dfyjoin.show(10) </pre>

10. Write Down Data from a Dynamic Frame using Glue Data Catalog	
	<pre># write data from the dynamicFrameCustomers to customers_write_dyf table using the meta data stored in the glue data catalog  glueContext.write_dynamic_frame.from_catalog(     frame=dynamicFrameCustomers,     database = "pyspark_tutorial_db",     table_name = "customers_write_dyf" )</pre>
11. Convert from Dynamic Frame To Spark DataFrame	
	<pre># dynamic Frame to Spark DataFrame sparkDf = dynamicFrameCustomers.toDF()  #show spark DF sparkDf.show()</pre>
12. Selecting columns in a spark dataframe	
13. Add Columns In A Spark Dataframe - creating a new column with a literal string	<pre># select columns from spark dataframe dfSelect = sparkDf.select("customerid","fullname")  # show selected dfSelect.show()</pre>
	<pre>#import lit from sql functions from pyspark.sql.functions import lit  # Add new column to spark dataframe dfNewColumn = sparkDf.withColumn("date", lit("2022-07-24"))  # show df with new column dfNewColumn.show()</pre>
14. Using concat to concatenate two columns	
	<pre># import concat from functions from pyspark.sql.functions import concat  # create another full name column dfNewFullName = sparkDf.withColumn("new_full_name",concat("firstname",concat(lit(' '), "lastname")))  # show full name column dfNewFullName.show()</pre>

15. Dropping Columns	<pre># drop column from spark dataframe dfDropCol = sparkDf.drop("firstname","lastname")  # show dropped column df dfDropCol.show()</pre>
16. Renaming columns	<pre># rename column in Spark dataframe dfRenameCol = sparkDf.withColumnRenamed("fullname","full_name")  # show renamed column dataframe dfRenameCol.show()</pre>
17. GroupBy and Aggregate functions	<pre># group by lastname then print counts of lastname sparkDf.groupBy("lastname").count().show()</pre>
18. Filtering columns and where clauses	<pre># filter spark DataFrame for customers who have the last name Adams sparkDf.filter(sparkDf["lastname"] == "Adams").show()  # where clause spark DataFrame for customers who have the last name Adams sparkDf.where("lastname == 'Adams'").show()</pre>
19. Joins - read the orders table and convert it to a spark dataframe	<pre># read from customers table in the glue data catalog using a dynamic frame convert to spark dataframe dfOrders = glueContext.create_dynamic_frame.from_catalog(     database = "pyspark_tutorial_db",     table_name = "orders" ).toDF()</pre>
20. Inner joins for spark dataframes	<pre># inner Join Customers Spark DF to Orders Spark DF sparkDf.join(dfOrders,sparkDf.customerid == dfOrders.customerid,"inner").show(truncate=False)</pre>
21. Inner joins for field members with a specified surname	<pre># inner join on Adams DF and orders dfAdams.join(dfOrders,dfAdams.customerid == dfOrders.customerid,"inner").show()</pre>

22. Left joins	<pre># left join on orders and adams df dfOrders.join(dfAdams,dfAdams.customerid == dfOrders.customerid,"left").show(100)</pre>
23. Writing data down using the glue data catalog	<pre># Import Dyanmic DataFrame class from awsglue.dynamicframe import DynamicFrame  #Convert from Spark Data Frame to Glue Dynamic Frame dyfCustomersConvert = DynamicFrame.fromDF(sparkDf, glueContext, "convert")  #Show converted Glue Dynamic Frame dyfCustomersConvert.show()</pre>
24. Writing data to an S3 location	<pre># write data from the converted dynamic frame to the S3 location. glueContext.write_dynamic_frame.from_options(     frame = dyfCustomersConvert,     connection_type="s3",     connection_options = {"path": "s3://&lt;YOUR_S3_BUCKET_NAME&gt;/write_down_dyf_to_s3"},     format = "csv",     format_options={         "separator": ",",     },     transformation_ctx = "datasink2")</pre>
25. Write data to a glue table using glue catalog metadata	<pre># write converted data to customers_write_dyf table using metadata stored in the glue data catalog glueContext.write_dynamic_frame.from_catalog(     frame = dyfCustomersConvert,     database = "pyspark_tutorial_db",     table_name = "customers_write_dyf")</pre>