P Y S P A R K N O T E B O O K

C O N T E N T S

01. Set up a job (inception	14. Using concat to concatenate two columns
02. Read from table / show 3 records	15. Dropping columns
03. Check data types	16. Renaming columns
04. ount The Number of rows in a dynamic dataframe	17. GroupBy and Aggregate functions
05. Select specific fields from a dynamic dataframe	18. Filtering columns and where clauses
06. Drop fields from a dynamic dataframe	19. Joins – read the orders table and convert to a SparkDf
07. Mapping an array, allow a column rename	20. Inner joins for spark dataframes
08. Filter a dynamic frame for customers by last name	21. Inner joins for field members with a specified surname
09. Join 2 dynamic frames using an equality join	22. Left joins
10. Write Down Data from a Dynamic Frame with Glue	23. Writing data down using the glue data catalog
11. Convert from Dynamic Frame To Spark DataFrame	24. Writing data to an S3 location
12. Selecting columns in a spark dataframe	25. Write data to a glue table using glue catalog metadata
13. Add Columns In A SparkDf, create a new column	

1.Set up a job (inception)

```
# 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)
```

2. Read from table / show 3 records

```
# 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)
```

3. Check data types	`# check types in dynamic frame` `dynamicFrameCustomers.printSchema()`
4. Count The Number of Rows in a dynamic dataframe	`# count The Number of Rows in a dyanamic dataframe` `dynamicFrameCustomers.count()`

5. Select specific fields from a dynamic dataframe	`# selecting certain fields from a dynamic dataframe`
	`dyfCustomerSelectFields = dynamicFrameCustomers.select_fields(["customerid", "fullname"])`
	
	`# show top 10`
	`dyfCustomerSelectFields.show(10)`

6. Drop fields from a dynamic dataframe	`# drop Fields of Dynamic Frame`
	`dyfCustomerDropFields = dynamicFrameCustomers.drop_fields(["firstname","lastname"])`
	
	`# show Top 10 rows of dyfCustomerDropFields Dynamic Frame`
	`dyfCustomerDropFields.show(10)`

7. Map an array to allow a column rename to take place	`# mapping array for column rename fullname -> name`
	`mapping=[("customerid", "long", "customerid","long"),("fullname", "string", "name", "string")]`
	
	`# apply the mapping to rename fullname -> name`
	`dfyMapping = ApplyMapping.apply(`
	`frame = dyfCustomerDropFields,`
	`mappings = mapping,`
	`transformation_ctx = "applymapping1"`
	`)`
	`# show the new dynamic frame with name column`
	`dfyMapping.show(10)`

8. Filter a dynamic frame for customers by last name	`# 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)`

9. Join 2 dynamic frames using an equality join	`# 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)`

10. Write Down Data from a Dynamic Frame using Glue	`# 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")`

11. Convert from Dynamic Frame To Spark DataFrame	`# dynamic Frame to Spark DataFrame`
	`sparkDf = dynamicFrameCustomers.toDF()`
	
	`#show spark DF`
	`sparkDf.show()`

12. Selecting columns in a spark dataframe	`# select columns from spark dataframe` `dfSelect = sparkDf.select("customerid","fullname")`

13. Add Columns In A SparkDf, create a new column	`# show selected`
	`dfSelect.show()`
	
	`#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()`

14. Using concat to concatenate two columns	`# 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()`

15. Dropping Columns	`# drop column from spark dataframe`
	`dfDropCol = sparkDf.drop("firstname","lastname")`
	
	`# show dropped column df`
	`dfDropCol.show()`

16. Renaming columns	`# rename column in Spark dataframe`
	`dfRenameCol = sparkDf.withColumnRenamed("fullname","full_name")`
	`# show renamed column dataframe`
	`dfRenameCol.show()`

17. GroupBy and Aggregate functions	`# group by lastname then print counts of lastname` `sparkDf.groupBy("lastname").count().show()`

18. Filtering columns and where clauses	`# 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()`

19. Joins, read the orders , convert it to a SparkDf	`# read from customers table in the glue data catalog using a dynamic frame convert to SparkDf`
	`dfOrders = glueContext.create_dynamic_frame.from_catalog(`
	`database = "pyspark_tutorial_db",`
	`table_name = "orders").toDF()`

20. Inner joins for spark dataframes	`# inner Join Customers Spark DF to Orders Spark DF` `sparkDf.join(dfOrders,sparkDf.customerid ==  dfOrders.customerid,"inner").show(truncate=False)`
21. Inner joins, field members with a specified surname	`# inner join on Adams DF and orders` `dfAdams.join(dfOrders,dfAdams.customerid ==  dfOrders.customerid,"inner").show()`
22. Left joins	`# left join on orders and adams df` `dfOrders.join(dfAdams,dfAdams.customerid ==  dfOrders.customerid,"left").show(100)`

23. Writing data down using the glue data catalog	`# 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()`

24. Writing data to an S3 location	`# 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://<YOUR_S3_BUCKET_NAME>/write_down_dyf_to_s3"},`
	`format = "csv",`
	`format_options={`
	`"separator": ","`
	`},`
	`transformation_ctx = "datasink2")`

25. Write data to a glue table using glue metadata	`# 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")`