**Azure Apache SparkPool NoteBook Queries written in PySpark**

1. PySpark query to create dataframe and read data

courses = [(1,'AZ-900',10.99),(2,'DP-900',10.99),(3,'DP-203',15.99)]

df=spark.createDataFrame(courses,['ID','CourseName','Price'])

display(df)

1. PySpark query to sort column based on particular column.

from pyspark.sql.functions import col

sorteddf=df.sort(col('CourseName').desc())

display(sorteddf)

1. PySpark query to add new row into the dataframe

new\_row = spark.createDataFrame([(4,'DP-102', 20.99)], ['ID','CourseName','Price'])

df=df.union(new\_row)

display(df)

1. PySpark query to load data from a file

%%pyspark

df = spark.read.load('abfss://csv@datalakecmak5p6.dfs.core.windows.net/2019.csv', format='csv')

/\* abfss means Azure Blob file Storage System

* ABFSS protocol is optimized for accessing data stored in Azure Blob Storage, including Azure Data Lake Storage Gen2. It provides better performance and efficiency compared to HTTPS.
* Integration with Spark: Apache Spark has built-in support for reading data from various file systems, including ABFSS. This native support ensures seamless integration and efficient data processing within Spark.
* Security and Authentication: ABFSS provides better integration with Azure's authentication mechanisms, such as Azure Active Directory (Azure AD) and Shared Key authentication. This allows you to securely access data in Azure Data Lake Storage Gen2 with proper authentication and authorization.

## If header exists uncomment line below and add after format parameter

## header=True \*/

display(df.limit(10))

1. PySpark query to select only few columns from DataFrame

from pyspark.sql.functions import col

df1.select(col("\_c2"),col("\_c3")).show()

1. PySpark query to filter based on column values of other dataframe

from pyspark.sql.functions import col

Christy\_zhu=df.filter(col("\_c3")=="Christy Zhu")

display(Christy\_zhu)

1. PySpark query to define schema

%%PySpark # Magic operator

from pyspark.sql.types import \*

from pyspark.sql.functions import \*

orderSchema = StructType([

StructField("SalesOrderNumber", StringType()),

StructField("SalesOrderLineNumber", IntegerType()),

StructField("OrderDate", DateType()),

StructField("CustomerName", StringType()),

StructField("Email", StringType()),

StructField("Item", StringType()),

StructField("Quantity", IntegerType()),

StructField("UnitPrice", FloatType()),

StructField("Tax", FloatType())

])

df = spark.read.load('abfss://files@datalakexxxxxxx.dfs.core.windows.net/sales/orders/\*.csv', format='csv', schema=orderSchema)

display(df.limit(100))

1. PySpark query to print schema

df.printSchema()

1. PySpark query to showcase Aggegartion functions

Christy\_zhu = df['CustomerName', 'Email'] ## new dataframe is getting created from already existing dataframe

print(Christy\_zhu.count())

print(Christy\_zhu.distinct().count())

display(Christy\_zhu.distinct())

customers = df.select("CustomerName", "Email").where(df['Item']=='Road-250 Red, 52')

print(customers.count())

print(customers.distinct().count())

display(customers.distinct())

productSales = df.select("Item", "Quantity").groupBy("Item").sum()

display(productSales)

yearlySales = df.select(year("OrderDate").alias("Year")).groupBy("Year").count().orderBy("Year")

display(yearlySales)

1. PySpark query to create view

df.createOrReplaceTempView("salesorders")

spark\_df = spark.sql("SELECT \* FROM salesorders")

display(spark\_df)