1. Read a CSV and print output to console?

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ReadCSV").getOrCreate()

df = spark.read.csv("path/to/your/file.csv", header=True, inferSchema=True)

df.show(truncate=False)

2.Read a JSON file and print output to console?

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ReadJSON").getOrCreate()

df = spark.read.json("path/to/your/file.json")

df.show(truncate=False) # 'truncate=False' ensures full column values are displayed

3.Read parquet file and print output to console?

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ReadParquet").getOrCreate()

df = spark.read.parquet("path/to/your/file.parquet")

df.show(truncate=False) # 'truncate=False' ensures full column values are displayed

4. Read a avro file and print output to console?

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ReadAvro") \

.config("spark.jars.packages", "org.apache.spark:spark-avro\_2.12:3.5.0") \

.getOrCreate()

df = spark.read.format("avro").load("path/to/your/file.avro")

df.show(truncate=False) # 'truncate=False' ensures full column values are displayed

5. Example for broadcast join (Inner join 2 dataframes)

from pyspark.sql import SparkSession

from pyspark.sql.functions import broadcast

spark = SparkSession.builder.appName("BroadcastJoinExample").getOrCreate()

data1 = [(1, "Alice"), (2, "Bob"), (3, "Charlie")]

df1 = spark.createDataFrame(data1, ["id", "name"])

data2 = [(1, "HR"), (2, "Finance")]

df2 = spark.createDataFrame(data2, ["id", "department"])

joined\_df = df1.join(broadcast(df2), "id", "inner")

joined\_df.show()

6. Example for Filtering the data

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("FilterExample").getOrCreate()

data = [

(1, "Alice", 25, "HR"),

(2, "Bob", 30, "Finance"),

(3, "Charlie", 35, "IT"),

(4, "David", 40, "HR"),

]

df = spark.createDataFrame(data, ["id", "name", "age", "department"])

filtered\_df = df.filter(df.age > 30)

filtered\_df.show()

7. Example for applying aggregate functions like max, min, avg

from pyspark.sql import SparkSession

from pyspark.sql.functions import max, min, avg

spark = SparkSession.builder.appName("AggregationExample").getOrCreate()

data = [

(1, "Alice", 25, "HR"),

(2, "Bob", 30, "Finance"),

(3, "Charlie", 35, "IT"),

(4, "David", 40, "HR"),

]

df = spark.createDataFrame(data, ["id", "name", "age", "department"])

agg\_df = df.agg(

max("age").alias("Max\_Age"),

min("age").alias("Min\_Age"),

avg("age").alias("Avg\_Age")

)

agg\_df.show()

8. Example for Read json file with typed schema (without infering the schema) with Structtype, StructField .....?

from pyspark.sql import SparkSession

from pyspark.sql.types import StructType, StructField, IntegerType, StringType

spark = SparkSession.builder.appName("ReadJSONWithSchema").getOrCreate()

json\_schema = StructType([

StructField("id", IntegerType(), True),

StructField("name", StringType(), True),

StructField("age", IntegerType(), True),

StructField("department", StringType(), True)

])

df = spark.read.schema(json\_schema).json("path/to/your/file.json")

df.show()

9. Example for increase and decrease number of dataframe partitions?

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("PartitionsExample").getOrCreate()

data = [(1, "Alice"), (2, "Bob"), (3, "Charlie"), (4, "David"), (5, "Eve")]

df = spark.createDataFrame(data, ["id", "name"])

print(f"Initial number of partitions: {df.rdd.getNumPartitions()}")

df\_repartitioned = df.repartition(4)

print(f"Number of partitions after repartitioning to 4: {df\_repartitioned.rdd.getNumPartitions()}")

df\_coalesced = df\_repartitioned.coalesce(2)

print(f"Number of partitions after coalescing to 2: {df\_coalesced.rdd.getNumPartitions()}")

10. Example for renaming the column of the dataframe

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("RenameColumnExample").getOrCreate()

data = [(1, "Alice"), (2, "Bob"), (3, "Charlie")]

df = spark.createDataFrame(data, ["id", "name"])

print("Original DataFrame:")

df.show()

df\_renamed = df.withColumnRenamed("name", "full\_name")

print("DataFrame after renaming the column:")

df\_renamed.show()

11. Example for adding a new column to the dataframe

from pyspark.sql import SparkSession

from pyspark.sql.functions import lit

spark = SparkSession.builder.appName("AddColumnExample").getOrCreate()

data = [(1, "Alice", 25), (2, "Bob", 30), (3, "Charlie", 35)]

df = spark.createDataFrame(data, ["id", "name", "age"])

print("Original DataFrame:")

df.show()

df\_with\_new\_column = df.withColumn("is\_adult", df["age"] >= 18)

print("DataFrame after adding the new column:")

df\_with\_new\_column.show()

df\_with\_constant\_column = df.withColumn("country", lit("USA"))

print("DataFrame after adding constant column:")

df\_with\_constant\_column.show()

12. Changing the structure of the dataframe

sample json:

[

"""{ "name" : "john doe", "dob" : "01-01-1980" }""",

"""{ "name" : "john adam", "dob" : "01-01-1960", "phone" : 1234567890 }"""

]

A. Comeup with your own data , add more data points

B. I want you to read the above dataset into dataframe

C. Now, change the dataframe structure and write the dataframe as json file

Json file structure should look like

{

{ "personal\_data" : { "name" : "john doe", "dob" : "01-01-1980" } } ,

{ "personal\_data" : { "name" : "john adam", "dob" : "01-01-1960", "phone" : 1234567890 } }

}

hint: read string as rdd , use struct function on dataframe

Ans)

from pyspark.sql import SparkSession

from pyspark.sql.functions import struct

from pyspark.sql.types import StructType, StructField, StringType, LongType

spark = SparkSession.builder.appName("ChangeStructureExample").getOrCreate()

data = [

"""{ "name" : "john doe", "dob" : "01-01-1980" }""",

"""{ "name" : "john adam", "dob" : "01-01-1960", "phone" : 1234567890 }""",

"""{ "name" : "jane smith", "dob" : "15-03-1990", "phone" : 9876543210 }""",

"""{ "name" : "bob brown", "dob" : "20-07-1975" }"""

]

rdd = spark.sparkContext.parallelize(data)

df = spark.read.json(rdd)

print("Original DataFrame:")

df.show(truncate=False)

# We'll create a new column 'personal\_data' as a struct containing 'name', 'dob', and 'phone' (if available)

df\_transformed = df.select(

struct("name", "dob", "phone").alias("personal\_data")

)

print("Transformed DataFrame:")

df\_transformed.show(truncate=False)

df\_transformed.write.mode("overwrite").json("path/to/output/transformed\_data.json")

13. Read from kafka source and print output as Stream

from confluent\_kafka import Consumer, KafkaException, KafkaError

conf = {

'bootstrap.servers': 'localhost:9092', # Kafka server address

'group.id': 'my\_consumer\_group', # Consumer group ID

'auto.offset.reset': 'earliest' # Read messages from the earliest offset

}

# Create a Kafka Consumer

consumer = Consumer(conf)

# Subscribe to the Kafka topic

topic = 'your\_topic\_name' # Replace with your Kafka topic

consumer.subscribe([topic])

print(f"Consuming messages from topic: {topic}")

# Consume messages in a loop

try:

while True:

msg = consumer.poll(timeout=1.0) # Poll for a message

if msg is None:

continue # No message available within timeout

if msg.error():

if msg.error().code() == KafkaError.\_PARTITION\_EOF:

# End of partition

print(f"End of partition reached {msg.partition} at offset {msg.offset()}")

else:

raise KafkaException(msg.error())

else:

# Message successfully consumed

print(f"Received message: {msg.value().decode('utf-8')}")

except KeyboardInterrupt:

print("Consumer interrupted")

finally:

# Close the consumer

consumer.close()