Pyspark Practice Question

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Pyspark_Practice") \
    .master("local[*]") \
    .getOrCreate()

spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.5.1
Master
local[*]
AppName
Pyspark_Practice

Basic RDD Operation

```
# initialise spark context
sc =  spark.sparkContext


# create an RDD from a list
rdd = sc.parallelize([1,2,3,4,5])


# perform a simple action to collect the data
rdd.collect()
```

```
[1, 2, 3, 4, 5]
```

```
# count element in rdd
rdd.count()
```

```
5
```

```
# filter even number in rdd
rdd.filter(lambda x:x%2==0).collect()
```

```
[2, 4]
```

```
# find square of rdd element
rdd.map(lambda x:x**2).collect()
```

```
[1, 4, 9, 16, 25]
```

```
# get distinct element in rdd
rdd.distinct().collect()
```

```
[2, 4, 1, 3, 5]
```

```
# reduce the rdd by summing all element

rdd.reduce(lambda x,y:x+y)
```

```
15
```

```
# union of two rdd
rdd2 = sc.parallelize([5,6,7,8,9])
rdd.union(rdd2).collect()
```

```
[1, 2, 3, 4, 5, 5, 6, 7, 8, 9]
```

```
# intersection of two rdd
rdd.intersection(rdd2).collect()
```

```
[5]
```

```
# group by key
rdd3 = sc.parallelize([("a",1),("b",2),("a",3),("b",4)])
rdd3.groupByKey().mapValues(list).collect()
```

```
[('b', [2, 4]), ('a', [1, 3])]
```

```
# reduce by key to sum all values
rdd3.reduceByKey(lambda a,b:a+b).collect()
```

```
[('b', 6), ('a', 4)]
```

🔹 DataFrame Basics

```
# create dataframe from list of tuples
data = [('Alice',1),('Bob',2),('Charlie',3)]
df = spark.createDataFrame(data,['name','age'])
df.show()
```

```
+-------+---+
|   name|age|
+-------+---+
|  Alice|  1|
|    Bob|  2|
|Charlie|  3|
+-------+---+
```

```
# show schema
df.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- age: long (nullable = true)
```

```
# select a column
df.select('Name').show()
```

```
+-------+
|   Name|
+-------+
|  Alice|
|    Bob|
|Charlie|
+-------+
```

```
#Filter DataFrame rows
df.select('Name').filter(df['age']>2).show()
```

```
+-------+
|   Name|
+-------+
|Charlie|
+-------+
```

```
df.filter(df['age']==3).show()
```

```
+-------+---+
|   name|age|
+-------+---+
|Charlie|  3|
+-------+---+
```

```
#Add new column
df.withColumn('double_age',df['age']*2).show()
```

```
+-------+---+----------+
|   name|age|double_age|
+-------+---+----------+
|  Alice|  1|         2|
|    Bob|  2|         4|
|Charlie|  3|         6|
+-------+---+----------+
```

```
# show dataframe
df.show()
```

```
+-------+---+
|   name|age|
+-------+---+
|  Alice|  1|
|    Bob|  2|
|Charlie|  3|
+-------+---+
```

```
# show new column with triple age
df = df.withColumn('triple_age',df['age']*3)
df.show()
```

```
+-------+---+----------+
|   name|age|triple_age|
+-------+---+----------+
|  Alice|  1|         3|
|    Bob|  2|         6|
|Charlie|  3|         9|
+-------+---+----------+
```

```
# drop a column
df.drop('triple_age').show()
```

```
+-------+---+
|   name|age|
+-------+---+
|  Alice|  1|
|    Bob|  2|
|Charlie|  3|
+-------+---+
```

```
# rename a column
df.withColumnRenamed('age','Age').show()
```

```
+-------+---+----------+
|   name|Age|triple_age|
+-------+---+----------+
|  Alice|  1|         3|
|    Bob|  2|         6|
|Charlie|  3|         9|
+-------+---+----------+
```

```
# sort by column
df.sort('Age',ascending = False).show()
```

```
+-------+---+----------+
|   name|age|triple_age|
+-------+---+----------+
|Charlie|  3|         9|
|    Bob|  2|         6|
|  Alice|  1|         3|
+-------+---+----------+
```

```
# create dataframe from list of tuples
data = [('Alice',1),('Bob',2),('Charlie',3),('Charlie',3),('Bob',4)]
df = spark.createDataFrame(data,['name','age'])
df.show()
```

```
+-------+---+
|   name|age|
+-------+---+
|  Alice|  1|
|    Bob|  2|
|Charlie|  3|
|Charlie|  3|
|    Bob|  4|
+-------+---+
```

```
# Group by and aggregate

df.groupBy('Name').agg({"Age":"sum"}).show()
```

```
+-------+--------+
|   Name|sum(Age)|
+-------+--------+
|    Bob|       6|
|  Alice|       1|
|Charlie|       6|
+-------+--------+
```

```
# get distinct rows
df.distinct().show()
```

```
+-------+---+
|   name|age|
+-------+---+
|    Bob|  2|
|  Alice|  1|
|Charlie|  3|
|    Bob|  4|
+-------+---+
```

🔹 SQL with DataFrames

```
# Register DataFrame as SQL table
df.createOrReplaceTempView("people")
```

```
# Run a sql query in sparkSQL
spark.sql("Select * from people where age > 2").show()
```

```
+-------+---+
|   name|age|
+-------+---+
|Charlie|  3|
|Charlie|  3|
|    Bob|  4|
+-------+---+
```

```
# count rows using sparkSQL
spark.sql("select count(1) as row_count from people").show()
```

```
+---------+
|row_count|
+---------+
|        5|
+---------+
```

```
# case statemetns in sparkSQL
spark.sql("select name, age, case when age > 2 then 'old' else 'young' end as age_group from people").show()
```

```
+-------+---+---------+
|   name|age|age_group|
+-------+---+---------+
|  Alice|  1|    young|
|    Bob|  2|    young|
|Charlie|  3|      old|
|Charlie|  3|      old|
|    Bob|  4|      old|
+-------+---+---------+
```

```
# join two dataframes
df2 = spark.createDataFrame([('Alice',1),('Bob',2)],['name','new_age'])
df2.show()
```

```
+-----+-------+
| name|new_age|
+-----+-------+
|Alice|      1|
|  Bob|      2|
+-----+-------+
```

```
# join dataframes on one column
df = df.join(df2, on='name', how='inner')
df.show()
```

```
+-----+---+-------+
| name|age|new_age|
+-----+---+-------+
|Alice|  1|      1|
|  Bob|  2|      2|
|  Bob|  4|      2|
+-----+---+-------+
```

🔹 Advanced Transformations

```
# Explode array column(transpose)
from pyspark.sql.functions import explode, col
df1 = spark.createDataFrame([([1,2,3],)], ["numbers"])


df1.select(explode(col("numbers"))).show()
```

```
+---+
|col|
+---+
|  1|
|  2|
|  3|
+---+
```

Note : If your df2 DataFrame has a column with arrays (e.g., column "letters"), you should do:

```
data = [("Alice", ["a", "b", "c"]), ("Bob", ["x", "y"])]
df2 = spark.createDataFrame(data, ["name", "letters"])

# Explode array column
df2.select("name", explode("letters").alias("letter")).show()
```

```
+-----+------+
| name|letter|
+-----+------+
|Alice|     a|
|Alice|     b|
|Alice|     c|
|  Bob|     x|
|  Bob|     y|
+-----+------+
```

```
# pivot dataframe
df.groupBy('name').pivot("age").sum("age").show()
```

```
+-----+----+----+----+
| name|   1|   2|   4|
+-----+----+----+----+
|Alice|   1|NULL|NULL|
|  Bob|NULL|   2|   4|
+-----+----+----+----+
```

```
# pivot with aggregation
df.groupBy('name').pivot('age').sum('new_age').show()
```

```
+-----+----+----+----+
| name|   1|   2|   4|
+-----+----+----+----+
|Alice|   1|NULL|NULL|
|  Bob|NULL|   2|   2|
+-----+----+----+----+
```

```
# window function
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number


# Sample data
data = [
    ("A", 90),
    ("A", 85),
    ("A", 92),
    ("B", 70),
    ("B", 88),
    ("B", 75)
]

# Create DataFrame
df = spark.createDataFrame(data, ["group", "score"])


# apply window function to assign row numbers within each group based on score
windowSpec = Window.partitionBy("group").orderBy(df['score'].desc())
df.withColumn("row_num", row_number().over(windowSpec)).show()
```

```
+-----+-----+-------+
|group|score|row_num|
+-----+-----+-------+
|    A|   92|      1|
|    A|   90|      2|
|    A|   85|      3|
|    B|   88|      1|
|    B|   75|      2|
|    B|   70|      3|
+-----+-----+-------+
```

```
# Aggregate with multiple functions
from pyspark.sql.functions import min, max
df.agg(min("score").alias('min_Score'), max("score").alias('max_Score')).show()
```

```
+---------+---------+
|min_Score|max_Score|
+---------+---------+
|       70|       92|
+---------+---------+
```

```
# Add literal column
from pyspark.sql.functions import lit
df.withColumn("country", lit("USA")).show()
```

```
+-----+-----+-------+
|group|score|country|
+-----+-----+-------+
|    A|   90|    USA|
|    A|   85|    USA|
|    A|   92|    USA|
|    B|   70|    USA|
|    B|   88|    USA|
|    B|   75|    USA|
+-----+-----+-------+
```

🔹 Performance and Optimization

```
# Cache a DataFrame
df.cache()
```

```
DataFrame[group: string, score: bigint]
```

```
# Persist with different storage level
from pyspark import StorageLevel
df.persist(StorageLevel.MEMORY_AND_DISK)
```

```
DataFrame[group: string, score: bigint]
```

```
df2 = df.withColumn('group',lit('A'))
```

```
# Broadcast join
from pyspark.sql.functions import broadcast
df.join(broadcast(df2), "group").show()
```

```
+-----+-----+-----+
|group|score|score|
+-----+-----+-----+
|    A|   90|   90|
|    A|   90|   85|
|    A|   90|   92|
|    A|   90|   70|
|    A|   90|   88|
|    A|   90|   75|
|    A|   85|   90|
|    A|   85|   85|
|    A|   85|   92|
|    A|   85|   70|
|    A|   85|   88|
|    A|   85|   75|
|    A|   92|   90|
|    A|   92|   85|
|    A|   92|   92|
|    A|   92|   70|
|    A|   92|   88|
|    A|   92|   75|
+-----+-----+-----+
```

```
# Repartition DataFrame
df.repartition(5)
```

```
DataFrame[group: string, score: bigint]
```

```
# Coalesce partitions
df.coalesce(1)
```

```
DataFrame[group: string, score: bigint]
```

🔹 File I/O

```
# read a csv file
uber_df = spark.read.csv("/content/uber_data.csv", header=True, inferSchema=True)
uber_df.show(5)
```

```
+--------+--------------------+--------------------+---------------+-------------+------------------+------------------+----------+----
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|  pickup_longitude|   pickup_latitude|RatecodeID|stor
+--------+--------------------+--------------------+---------------+-------------+------------------+------------------+----------+----
|       1| 2016-03-01 00:00:00|  2016-03-01 00:07:55|              1|          2.5|-73.97674560546875|40.765151977539055|         1|
|       1| 2016-03-01 00:00:00|  2016-03-01 00:11:06|              1|          2.9|-73.98348236083984|40.767925262451165|         1|
|       2| 2016-03-01 00:00:00|  2016-03-01 00:31:06|              2|        19.98|-73.78202056884764| 40.64480972290039|         1|
|       2| 2016-03-01 00:00:00|  2016-03-01 00:00:00|              3|        10.78|-73.86341857910156|40.769813537597656|         1|
|       2| 2016-03-01 00:00:00|  2016-03-01 00:00:00|              5|        30.43|-73.97174072265625| 40.79218292236328|         3|
+--------+--------------------+--------------------+---------------+-------------+------------------+------------------+----------+----
only showing top 5 rows
```

```
# sample json store in file sample_test_json
sample_json = [{"name": "Alice", "age": 30, "city": "New York"},
  {"name": "Bob", "age": 25, "city": "San Francisco"},
```

```
  {"name": "Charlie", "age": 35, "city": "Chicago"}]
```

```
# read from a json
sample_df = spark.read.json('/content/sample_test_json.json')
sample_df.show()
```

```
+---+-------------+-------+
|age|         city|   name|
+---+-------------+-------+
| 30|     New York|  Alice|
| 25|San Francisco|    Bob|
| 35|      Chicago|Charlie|
+---+-------------+-------+
```

```
# Write DataFrame to Parquet
uber_df.write.mode('overwrite').parquet("/content/uber_output/")
```

```
# Write DataFrame as CSV
df.write.option("header", True).mode('overwrite').csv("/content/output_csv/")
```

```
# Read Parquet
parquet_df = spark.read.parquet("/content/uber_output/")
parquet_df.show(5)
```

```
+--------+--------------------+--------------------+---------------+-------------+------------------+------------------+----------+----
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|  pickup_longitude|   pickup_latitude|RatecodeID|stor
+--------+--------------------+--------------------+---------------+-------------+------------------+------------------+----------+----
|       1| 2016-03-01 00:00:00|  2016-03-01 00:07:55|              1|          2.5|-73.97674560546875|40.765151977539055|         1|
|       1| 2016-03-01 00:00:00|  2016-03-01 00:11:06|              1|          2.9|-73.98348236083984|40.767925262451165|         1|
|       2| 2016-03-01 00:00:00|  2016-03-01 00:31:06|              2|        19.98|-73.78202056884764| 40.64480972290039|         1|
|       2| 2016-03-01 00:00:00|  2016-03-01 00:00:00|              3|        10.78|-73.86341857910156|40.769813537597656|         1|
|       2| 2016-03-01 00:00:00|  2016-03-01 00:00:00|              5|        30.43|-73.97174072265625| 40.79218292236328|         3|
+--------+--------------------+--------------------+---------------+-------------+------------------+------------------+----------+----
only showing top 5 rows
```

🔹 Date and Time Handling

```
from pyspark.sql.functions import current_date

# current date
df = df.withColumn("today", current_date())
```

```
df.show()
```

```
+-----+-----+----------+
|group|score|     today|
+-----+-----+----------+
|    A|   90|2025-07-29|
|    A|   85|2025-07-29|
|    A|   92|2025-07-29|
|    B|   70|2025-07-29|
|    B|   88|2025-07-29|
|    B|   75|2025-07-29|
+-----+-----+----------+
```

```
# Extract year/month
from pyspark.sql.functions import year, month

df = df.withColumn("year", year("today"))
df.show()
```

```
+-----+-----+----------+----+
|group|score|     today|year|
+-----+-----+----------+----+
|    A|   90|2025-07-29|2025|
|    A|   85|2025-07-29|2025|
|    A|   92|2025-07-29|2025|
|    B|   70|2025-07-29|2025|
|    B|   88|2025-07-29|2025|
```

```
|    B|   75|2025-07-29|2025|
+-----+-----+----------+----+
```

```
# extract month
df = df.withColumn("month", month("today"))
df.show()
```

```
+-----+-----+----------+----+-----+
|group|score|     today|year|month|
+-----+-----+----------+----+-----+
|    A|   90|2025-07-29|2025|    7|
|    A|   85|2025-07-29|2025|    7|
|    A|   92|2025-07-29|2025|    7|
|    B|   70|2025-07-29|2025|    7|
|    B|   88|2025-07-29|2025|    7|
|    B|   75|2025-07-29|2025|    7|
+-----+-----+----------+----+-----+
```

```
# Add a constant column
df = df.withColumn("start", lit('2025-07-20'))
df.show()
```

```
+-----+-----+----------+----+-----+----------+
|group|score|     today|year|month|     start|
+-----+-----+----------+----+-----+----------+
|    A|   90|2025-07-29|2025|    7|2025-07-20|
|    A|   85|2025-07-29|2025|    7|2025-07-20|
|    A|   92|2025-07-29|2025|    7|2025-07-20|
|    B|   70|2025-07-29|2025|    7|2025-07-20|
|    B|   88|2025-07-29|2025|    7|2025-07-20|
|    B|   75|2025-07-29|2025|    7|2025-07-20|
+-----+-----+----------+----+-----+----------+
```

```
# Date difference
from pyspark.sql.functions import datediff
df = df.withColumn("days_diff", datediff("today", "start"))
df.show()
```

```
+-----+-----+----------+----+-----+----------+---------+
|group|score|     today|year|month|     start|days_diff|
+-----+-----+----------+----+-----+----------+---------+
|    A|   90|2025-07-29|2025|    7|2025-07-20|        9|
|    A|   85|2025-07-29|2025|    7|2025-07-20|        9|
|    A|   92|2025-07-29|2025|    7|2025-07-20|        9|
|    B|   70|2025-07-29|2025|    7|2025-07-20|        9|
|    B|   88|2025-07-29|2025|    7|2025-07-20|        9|
|    B|   75|2025-07-29|2025|    7|2025-07-20|        9|
+-----+-----+----------+----+-----+----------+---------+
```

```
# Add days to date
from pyspark.sql.functions import date_add
df = df.withColumn("next_day", date_add("today", 1))
df.show()
```

```
+-----+-----+----------+----+-----+----------+---------+----------+
|group|score|     today|year|month|     start|days_diff|  next_day|
+-----+-----+----------+----+-----+----------+---------+----------+
|    A|   90|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|
|    A|   85|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|
|    A|   92|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|
|    B|   70|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|
|    B|   88|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|
|    B|   75|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|
+-----+-----+----------+----+-----+----------+---------+----------+
```

```
# print schema
df.printSchema()
```

```
root
 |-- group: string (nullable = true)
 |-- score: long (nullable = true)
 |-- today: date (nullable = false)
 |-- year: integer (nullable = false)
```

```
 |-- month: integer (nullable = false)
 |-- start: string (nullable = false)
 |-- days_diff: integer (nullable = true)
 |-- next_day: date (nullable = false)
```

```
# Convert string date to date format
from pyspark.sql.functions import to_date
df = df.withColumn('start_date',to_date("start","yyyy-MM-dd"))
df.show()
```

```
+-----+-----+----------+----+-----+----------+---------+----------+----------+
|group|score|     today|year|month|     start|days_diff|  next_day|start_date|
+-----+-----+----------+----+-----+----------+---------+----------+----------+
|    A|   90|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|2025-07-20|
|    A|   85|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|2025-07-20|
|    A|   92|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|2025-07-20|
|    B|   70|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|2025-07-20|
|    B|   88|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|2025-07-20|
|    B|   75|2025-07-29|2025|    7|2025-07-20|        9|2025-07-30|2025-07-20|
+-----+-----+----------+----+-----+----------+---------+----------+----------+
```

```
# print schema to confirm date format changes
df.printSchema()
```

```
root
 |-- group: string (nullable = true)
 |-- score: long (nullable = true)
 |-- today: date (nullable = false)
 |-- year: integer (nullable = false)
 |-- month: integer (nullable = false)
 |-- start: string (nullable = false)
 |-- days_diff: integer (nullable = true)
 |-- next_day: date (nullable = false)
 |-- start_date: date (nullable = true)
```

🔹 Miscellaneous

```
# check spark version
spark.version
```

```
'3.5.1'
```

```
# Convert RDD to DataFrame
rdd = sc.parallelize([(1,'ALICE'),(2,'BOB'),(3,None)])
df_new = rdd.toDF(["id","name"])
df_new.show()
```

```
+---+-----+
| id| name|
+---+-----+
|  1|ALICE|
|  2|  BOB|
|  3| NULL|
+---+-----+
```

```
# print first row of DataFrame
df_new.first()
```

```
Row(id=1, name='ALICE')
```

```
# check for null values in a column
from pyspark.sql.functions import col, isnull
df_new.filter(isnull(col("name"))).count()
```

```
1
```

```
spark.stop()
```