1. Write a program to create a RDD with 5 partitions for below array and print the number of partitions

|  |
| --- |
| data = [1,2,3,4,5,6,7,8,9,10,11,12] |

Solution🡪

from pyspark.sql import SparkSession

# Initialize SparkSession

spark = SparkSession.builder.master("local[1]").appName("RDD Partitions").getOrCreate()

# Create RDD with 5 partitions

data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

rdd = spark.sparkContext.parallelize(data, 5)

# Print number of partitions

print("Number of partitions is: " + str(rdd.getNumPartitions()))

2. Copy below data in a text file and create an RDD and count the number of records

|  |
| --- |
| 1827|2008-01-01|WE|1|JAN|1|2008|TRUE  1828|2008-01-02|TH|1|JAN|1|2008|FALSE  1829|2008-01-03|FR|1|JAN|1|2008|FALSE  1830|2008-01-04|SA|2|JAN|1|2008|FALSE  1831|2008-01-05|SU|2|JAN|1|2008|FALSE  1832|2008-01-06|MO|2|JAN|1|2008|FALSE  1833|2008-01-07|TU|2|JAN|1|2008|FALSE  1834|2008-01-08|WE|2|JAN|1|2008|FALSE  1835|2008-01-09|TH|2|JAN|1|2008|FALSE  1836|2008-01-10|FR|2|JAN|1|2008|FALSE  1837|2008-01-11|SA|3|JAN|1|2008|FALSE  1838|2008-01-12|SU|3|JAN|1|2008|FALSE  1839|2008-01-13|MO|3|JAN|1|2008|FALSE  1840|2008-01-14|TU|3|JAN|1|2008|FALSE  1841|2008-01-15|WE|3|JAN|1|2008|FALSE  1842|2008-01-16|TH|3|JAN|1|2008|FALSE  1843|2008-01-17|FR|3|JAN|1|2008|FALSE  1844|2008-01-18|SA|4|JAN|1|2008|FALSE  1845|2008-01-19|SU|4|JAN|1|2008|FALSE  1846|2008-01-20|MO|4|JAN|1|2008|FALSE  1847|2008-01-21|TU|4|JAN|1|2008|FALSE  1848|2008-01-22|WE|4|JAN|1|2008|FALSE  1849|2008-01-23|TH|4|JAN|1|2008|FALSE  1850|2008-01-24|FR|4|JAN|1|2008|FALSE  1851|2008-01-25|SA|5|JAN|1|2008|FALSE  1852|2008-01-26|SU|5|JAN|1|2008|FALSE |

Solution🡪

from pyspark.sql import SparkSession

# Initialize Spark Session

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

# Read data into RDD

rdd = spark.sparkContext.textFile("file:///home/takeo/data/spark\_assessment2/sample\_data.txt")

# Count the number of records

record\_count = rdd.count()

print("Number of records:", record\_count)

3. Write a program to assign a count to each word given in the paragraph below.

|  |
| --- |
| Python Lists allow us to hold items of heterogeneous types. In this article, we will learn how to create a list in Python; access the list items; find the number of items in the list, how to add an item to list; how to remove an item from the list; loop through list items; sorting a list, reversing a list; and many more transformation and aggregation actions on Python Lists. |

Expected output should look like

|  |
| --- |
| ('Python', 2)  ('Lists', 1)  ('allow', 1)  ('us', 1)  ('to', 5)  ('hold', 1)  ('items', 2)  ('of', 2)  ('heterogeneous', 1) |

Solution🡪

# Split into words, create RDD, and count occurrences

rdd1 = paragraph.split()

rdd2 = spark.sparkContext.parallelize(rdd1)

rdd3 = rdd2.map(lambda x: (x,1))

rdd4 = rdd3.reduceByKey(lambda a,b: a+b)

# Print word counts

for x,y in rdd4.collect():

print(f"('{x}', {y})")

4. Write a program and sql for followings

1. Register dataframe as view named as **Users**
2. **Return the details for only those users where salary is more than 3000**

|  |
| --- |
| data = [("James","","Smith","36636","M",3000),  ("Michael","Rose","","40288","M",4000),  ("Robert","","Williams","42114","M",4000),  ("Maria","Anne","Jones","39192","F",4000),  ("Jen","Mary","Brown","","F",-1)  ]  schema = StructType([ \  StructField("firstname",StringType(),True), \  StructField("middlename",StringType(),True), \  StructField("lastname",StringType(),True), \  StructField("id", StringType(), True), \  StructField("gender", StringType(), True), \  StructField("salary", IntegerType(), True) \  ]) |

**Solution🡪**

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

# Data and Schema

data = [

("James", "", "Smith", "36636", "M", 3000),

("Michael", "Rose", "", "40288", "M", 4000),

("Robert", "", "Williams", "42114", "M", 4000),

("Maria", "Anne", "Jones", "39192", "F", 4000),

("Jen", "Mary", "Brown", "", "F", -1)

]

schema = StructType([

StructField("firstname", StringType(), True),

StructField("middlename", StringType(), True),

StructField("lastname", StringType(), True),

StructField("id", StringType(), True),

StructField("gender", StringType(), True),

StructField("salary", IntegerType(), True)

])

# Create DataFrame and register as view

df = spark.createDataFrame(data, schema)

df.createOrReplaceTempView("Users")

# Query to get users where salary is more than 3000

sql\_query = spark.sql("SELECT \* FROM Users WHERE salary > 3000")

sql\_query.show()

**5.**  Write a program and sql for followings

1. Register dataframe as view named as **Users**
2. **Write a sql to return** 'firstname' when 'lastname' is Rose

|  |
| --- |
| structureData = [  (("James","","Smith"),"36636","M",3100),  (("Michael","Rose",""),"40288","M",4300),  (("Robert","","Williams"),"42114","M",1400),  (("Maria","Anne","Jones"),"39192","F",5500),  (("Jen","Mary","Brown"),"","F",-1)  ]  structureSchema = StructType([  StructField('name', StructType([  StructField('firstname', StringType(), True),  StructField('middlename', StringType(), True),  StructField('lastname', StringType(), True)  ])),  StructField('id', StringType(), True),  StructField('gender', StringType(), True),  StructField('salary', IntegerType(), True)  ]) |

**Soultion🡪**

structureData = [

(("James", "", "Smith"), "36636", "M", 3100),

(("Michael", "Rose", ""), "40288", "M", 4300),

(("Robert", "", "Williams"), "42114", "M", 1400),

(("Maria", "Anne", "Jones"), "39192", "F", 5500),

(("Jen", "Mary", "Brown"), "", "F", -1)

]

structureSchema = StructType([

StructField('name', StructType([

StructField('firstname', StringType(), True),

StructField('middlename', StringType(), True),

StructField('lastname', StringType(), True)

])),

StructField('id', StringType(), True),

StructField('gender', StringType(), True),

StructField('salary', IntegerType(), True)

])

# Create DataFrame and register as view

df = spark.createDataFrame(structureData, structureSchema)

df.createOrReplaceTempView("Users1")

# Query to return firstname when lastname is 'Rose'

sql\_query1 = spark.sql("SELECT name.firstname FROM Users1 WHERE name.lastname = 'Rose'")

sql\_query1.show()