**Pyspark Assignment**

**Name: Podutur Lahari - DE126**

**Date: 20-11-2024**

**Transformation & Actions in Pyspark**

**Transformations** are a kind of operation that takes an RDD as input and produces another RDD as output. Once a transformation is applied to an RDD, it returns a new RDD, the original RDD remains the same and thus are immutable. After applying the transformation, it creates a Directed Acyclic Graph or DAG for computations and ends after applying any actions on it. This is the reason they are called lazy evaluation processes.

**Actions** are a kind of operation which are applied on an RDD to produce a single value. These methods are applied on a resultant RDD and produces a non-RDD value, thus removing the laziness of the transformation of RDD.

**Actions:**

1. collect()
2. count()
3. first()
4. take()
5. reduce()

**Transformations:**

1. map()
2. filter()
3. union()
4. flatMap()
5. **The .collect() Action :**

**.collect() -** Retrieves all elements of the RDD and brings them to the driver as a list.  
**Use Case:** Useful for small datasets to gather results for further processing on the driver.  
**Example:**

#to create rdds and  dataframe

from pyspark import SparkContext

from pyspark.sql import  SparkSession

sc =SparkContext.getOrCreate()

spark = SparkSession.builder.appName('pyspark first program').getOrCreate()

#create the rdd

rdd = sc.parallelize([('C',85,76,87,91), ('B',85,76,87,91), ("A", 85,78,96,92), ("A", 92,76,89,96)], 4)

mydata = ['Division','English','Mathematics','Physics','Chemistry']

marks\_df = spark.createDataFrame(rdd, schema=mydata)

print(rdd.collect())

print(rdd) #---Transformation which gives rdd value

print(marks\_df.show())

rdd.collect() #----Action gives non rdd value

A screenshot of a computer

Description automatically generated

1. **The .count() Action :**

**.count() -** Returns the total number of elements in the RDD.  
**Use Case:** Helps determine the size of the dataset.  
**Example:**

from pyspark import SparkContext

sc = SparkContext.getOrCreate()

count\_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])

print(count\_rdd.count())

count\_rdd.count()

A screenshot of a phone

Description automatically generated

1. **The .first() Action :**

**.first() -** Returns the first element of the RDD.  
**Use Case:** Quick inspection of the initial value in the dataset.  
**Example:**

from pyspark import SparkContext

sc = SparkContext.getOrCreate()

first\_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])

first\_rdd.first() #First method is Action

**A screenshot of a computer

Description automatically generated**

1. **The .take() Action :**

**.take(n) -** Retrieves the first n elements of the RDD.  
**Use Case:** Useful for sampling data without returning the entire dataset.  
**Example:**

from pyspark import SparkContext

sc = SparkContext.getOrCreate()

count\_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])

print(count\_rdd.take(2))

count\_rdd.take(6)

A screenshot of a computer

Description automatically generated

1. **The .reduce() Action :**

**.reduce(func) -** Aggregates elements in the RDD using a binary function.  
**Use Case:** Commonly used for operations like summation or finding the maximum.  
**Example:**

from pyspark import SparkContext

sc = SparkContext.getOrCreate()

reduce\_rdd = sc.parallelize([1,2,3])

print(reduce\_rdd.reduce(lambda x,y:x+y))

A screenshot of a computer

Description automatically generated

1. **The .map() Transformation :**

**.map(func) -**Applies a function to each element in the RDD and returns a new RDD.  
**Use Case:** Transforms data without changing the structure. **Example:**

from pyspark import SparkContext

sc = SparkContext.getOrCreate()

map\_rdd = sc.parallelize([1,2,3])

print(map\_rdd.map(lambda x:x+10))

print(map\_rdd.map(lambda x:x+10).collect())

A screenshot of a chat

Description automatically generated

1. **The .filter() Transformation :**

**.filter(func) -** Retains elements that satisfy a given condition.  
**Use Case:** Extracts subsets of data based on criteria.  
**Example:**

from pyspark import SparkContext

sc = SparkContext.getOrCreate()

filter\_rdd = sc.parallelize([1,2,3])

print(filter\_rdd.filter(lambda x:x%2==0))

print(filter\_rdd.filter(lambda x:x%2==0).collect())

A screenshot of a computer

Description automatically generated

1. **The .union() Transformation :**

**.union(otherRdd) –** Combines two RDDs into one, containing all elements from both.  
**Use Case:** Merging datasets.  
**Example:**

union\_rdd=sc.parallelize([1,2,3,4,5,6,7,8,9,10])

union\_rdd1=union\_rdd.filter(lambda x: x%2==0)

union\_rdd2=union\_rdd.filter(lambda x: x%2!=0)

union\_rdd1.union(union\_rdd2).collect()

A screenshot of a phone

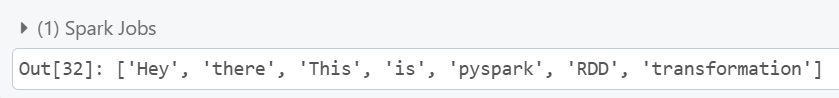
Description automatically generated

1. **The .flatMap() Transformation :**

**.flatMap(func) -** Similar to map, but each input element can produce multiple output elements.  
**Use Case:** Splits elements into smaller parts, such as words from a sentence.  
**Example:**

flatmap\_rdd=sc.parallelize(['Hey there', 'This is pyspark RDD transformation'])

(flatmap\_rdd.flatMap(lambda x: x.split(' ')).collect())



**Selecting, Renaming, Filtering Data in a Pandas DataFrame**

**1) .withColumnRenamed(existingName, newName) -** Renames a specific column in the DataFrame.

**Example:**

# Importing necessary libraries

from pyspark.sql import SparkSession

# Create a spark session

spark = SparkSession.builder.appName('pyspark - example join').getOrCreate()

# Create data in dataframe

data = [(('Ram'), '1991-04-01', 'M', 3000),(('Mike'), '2000-05-19', 'M', 4000),

(('Rohini'), '1978-09-05', 'M', 4000),(('Maria'), '1967-12-01', 'F', 4000),

(('Jenis'), '1980-02-17', 'F', 1200)]

# Column names in dataframe

columns = ["Name", "DOB", "Gender", "salary"]

# Create the spark dataframe

df = spark.createDataFrame(data=data,schema=columns)

df.withColumnRenamed("DOB","date of birth").show()

df.withColumnRenamed("DOB","date of birth").withColumnRenamed("Name","personname").show()

A screenshot of a computer

Description automatically generated

**2) .selectExpr(\*expressions) -** Allows SQL-style column aliasing and transformations in a DataFrame.

**Example:**

# Importing necessary libraries using select exp

from pyspark.sql import SparkSession

# Create a spark session

spark = SparkSession.builder.appName('pyspark - example join').getOrCreate()

# Create data in dataframe

data = [(('Ram'), '1991-04-01', 'M', 3000),

        (('Mike'), '2000-05-19', 'M', 4000),

        (('Rohini'), '1978-09-05', 'M', 4000),

        (('Maria'), '1967-12-01', 'F', 4000),

        (('Jenis'), '1980-02-17', 'F', 1200)]

# Column names in dataframe

columns = ["Name", "DOB", "Gender", "salary"]

# Create the spark dataframe

df = spark.createDataFrame(data=data,schema=columns)

data = df.selectExpr("Gender as category","DOB","Name as name","salary")

data.show()

A screenshot of a computer

Description automatically generated

**3) .col(columnName) -** Refers to a specific column in a DataFrame, useful for applying transformations or aliasing.

**.alias(newName) -** Renames a selected column in a DataFrame.

**Example:**

from pyspark.sql.functions import col

# Select the 'salary' as 'Amount' using aliasing

# Select remaining with their original name

data = df.select(col("Name"),col("DOB"),

                 col("Gender"),

                 col("salary").alias('Amount'))

data.show()

A screenshot of a computer

Description automatically generated

**Pyspark view and temp view**

**1) .createOrReplaceTempView("name") -** Registers the DataFrame as a temporary table (view) with a specified name, enabling SQL queries on it. The table exists only for the duration of the SparkSession.

**.toDF(columns) -** Converts an RDD into a DataFrame with specified column names.

**Example:**

from pyspark.sql import SparkSession

# Create spark session

spark = SparkSession \

.builder \

.appName("SparkByExamples.com") \

.enableHiveSupport() \

.getOrCreate()

data = [("James","Smith","USA","CA"),("Michael","Rose","USA","NY"),

("Robert","Williams","USA","CA"),("Maria","Jones","USA","FL")]

columns = ["firstname","lastname","country","state"]

# Create dataframe

sampleDF = spark.sparkContext.parallelize(data).toDF(columns)

sampleDF.createOrReplaceTempView("Person")

sampleDF.createOrReplaceTempView("mydata")

sampleDF.show()

A screenshot of a computer

Description automatically generated

**2) spark.sql("...") -** Executes an SQL query on a registered temporary view or Hive table, returning the result as a DataFrame.

**Example:**

spark.sql("select \* from person").show()

spark.sql("select \* from mydata").show()

A screenshot of a computer

Description automatically generated