**BDA Executed:**

**Program1:**

**#1**

import findspark

findspark.init()

# Create SparkSession and sparkcontext

from pyspark.sql import SparkSession

spark = SparkSession.builder\

.master("local")\

.appName('Firstprogram')\

.getOrCreate()

sc=spark.sparkContext

# Read the input file and Calculating words count

# Updated the file path to include the full path if the file is not in the current working directory

# Or ensure that the file is present in the current working directory

text\_file = sc.textFile("sample.txt") # Changed file path

counts = text\_file.flatMap(lambda line: line.split(" ")) \

.map(lambda word: (word, 1)) \

.reduceByKey(lambda x, y: x + y)

# Printing each word with its respective count

output = counts.collect()

for (word, counts) in output:

print("%s: %i" % (word, counts))

sc.stop()

spark.stop()

**Program 2:**

import findspark

findspark.init()

# Create SparkSession and sparkcontext

from pyspark.sql import SparkSession

spark = SparkSession.builder\

.master("local")\

.appName('Firstprogram')\

.getOrCreate()

sc=spark.sparkContext

#Using parallelize method, create RDD

rdd1=sc.parallelize([("spark", 1),("hadoop", 4)])

rdd2=sc.parallelize([("spark", 2),("hadoop", 5)])

#Perforn Join Operation on the created RDDs

rdd=sorted(rdd1.join(rdd2).collect())

#Print the Result

print(rdd)

rdd3=sorted(rdd1.fullOuterJoin(rdd2).collect())

print(rdd3)

print(sorted(rdd1.leftOuterJoin(rdd2).collect()))

rdd1 = sc.parallelize([("a", 1), ("b", 4)])

rdd2 = sc.parallelize([("a", 2), ("a", 3)])

sorted(rdd1.join(rdd2).collect())

rdd1 = sc.parallelize([("a", 1), ("b", 4)])

rdd2 = sc.parallelize([("a", 2)])

sorted(rdd1.leftOuterJoin(rdd2).collect())

rdd1 = sc.parallelize([("a", 1), ("b", 4)])

rdd2 = sc.parallelize([("a", 2)])

sorted(rdd2.rightOuterJoin(rdd1).collect())

rdd1 = sc.parallelize([("a", 1), ("b", 4)])

rdd2 = sc.parallelize([("a", 2), ("c", 8)])

sorted(rdd1.fullOuterJoin(rdd2).collect())

**Program 3 a):**

from pyspark import SparkContext

# create a Spark context

sc = SparkContext.getOrCreate()

# create an RDD of set of numbers

rdd = sc.parallelize([{1, 2, 3}, {4, 5, 6}, {7, 8, 9}])

# define an accumulator

acc = sc.accumulator(0)

# use the accumulator to sum the numbers in the RDD

def add\_to\_acc(x):

global acc

acc += sum(x)

rdd.foreach(add\_to\_acc)

print("Sum of numbers in RDD: ", acc.value)

# stop the Spark context

# sc.stop()

**Program 3b:**

from pyspark.sql import SparkSession

# create a SparkSession object

spark = SparkSession.builder.appName("CSV RDD").getOrCreate()

# create an RDD from a CSV file

rdd = spark.read.format("csv").option("header", "true").load("\sample\_data.csv").rdd

# display the top 5 rows of the RDD

print(rdd.take(5))

# convert the RDD to a DataFrame

df = rdd.toDF()

# display the statistical results

df.describe().show()

**Program 4th**import findspark

from pyspark.sql import SparkSession

from pyspark.sql import Row

from pyspark.sql.functions import desc

spark = SparkSession.builder\

.master("local")\

.appName('Firstprogram')\

.getOrCreate()

sc=spark.sparkContext

df = spark.createDataFrame([(2, "Alice"), (5, "Bob")]).toDF("age", "name")

df2 = spark.createDataFrame([Row(height=80, name="Tom"), Row(height=85, name="Bob")])

df3 = spark.createDataFrame([Row(age=2, name="Alice"), Row(age=5, name="Bob")])

df4 = spark.createDataFrame([

Row(age=10, height=80, name="Alice"),

Row(age=5, height=None, name="Bob"),

Row(age=None, height=None, name="Tom"),

Row(age=None, height=None, name=None),

])

df.join(df2, 'name').select(df.name, df2.height).show()

df.join(df2, df.name == df2.name, 'outer').select(

df.name, df2.height).sort(desc("name")).show()  
  
  
df.join(

df3,

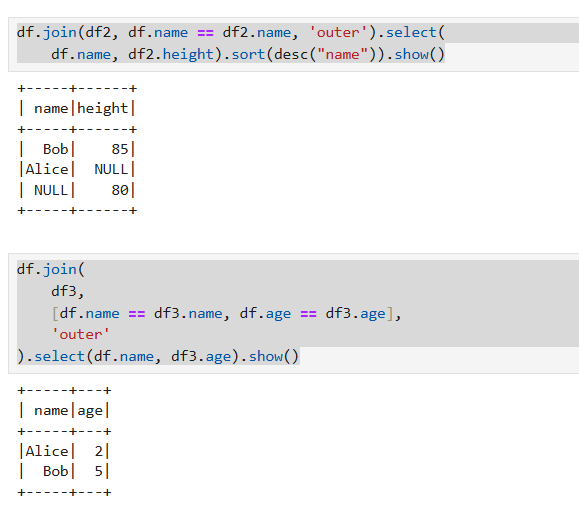
[df.name == df3.name, df.age == df3.age],

'outer'

).select(df.name, df3.age).show()

**Screenshot of executed program :**





**Program 5-1),2):**

from pyspark.sql import SparkSession

# Initialize Spark session

spark = SparkSession.builder \

.appName("RDD and DataFrame Example") \

.getOrCreate()

data = [("1", "john jones"), ("2", "tracey smith"), ("3", "amy sanders")]

columns = ["Seqno", "Name"]

rdd = spark.sparkContext.parallelize(data)

from pyspark.sql import Row

df = rdd.map(lambda x: Row(Seqno=x[0], Name=x[1])).toDF()

df.show()

**5-3),4)**

def capitalize\_first\_letter(s):

return s.title() if s else s

from pyspark.sql.functions import udf

from pyspark.sql.types import StringType

# Define a function to capitalize the first letter

def capitalize\_first\_letter(name):

return " ".join([word.capitalize() for word in name.split()])

capitalize\_udf = udf(capitalize\_first\_letter, StringType())

data = [("1", "john jones"), ("2", "tracey smith"), ("3", "amy sanders")]

columns = ["Seqno", "Name"]

df = spark.createDataFrame(data, columns)

df\_transformed = df.withColumn("Name", capitalize\_udf(df["Name"]))

df\_transformed.show()

**program 6:**

data = [("James","Sales","NY",90000,34,10000),

("Michael","Sales","NV",86000,56,20000),

("Robert","Sales","CA",81000,30,23000),

("Maria","Finance","CA",90000,24,23000),

("Raman","Finance","DE",99000,40,24000),

("Scott","Finance","NY",83000,36,19000),

("Jen","Finance","NY",79000,53,15000),

("Jeff","Marketing","NV",80000,25,18000),

("Kumar","Marketing","NJ",91000,50,21000)]

from pyspark.sql import SparkSession

from pyspark.sql import Row

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

# Initialize Spark session

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

# Define schema

schema = ["employee\_name","department","state","salary","age","bonus"]

# Create RDD

rdd = spark.sparkContext.parallelize(data)

# Convert RDD to DataFrame

df = spark.createDataFrame(rdd, schema)

# Group by state and sum the salaries

result = df.groupBy("state").sum("salary")

# Filter states where the total salary is greater than 1 lakh

filtered\_result = result.filter("sum(salary) > 100000")

# Sort by salary in descending order

final\_result = filtered\_result.orderBy("sum(salary)", ascending=False)

# Show the result

final\_result.show()

program 7:

from pyspark.sql import SparkSession

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

df = spark.createDataFrame(

[

("sue", 32),

("li", 3),

("bob", 75),

("heo", 13),

],

["first\_name", "age"],

)

df.show()

from pyspark.sql.functions import col, when

df1 = df.withColumn(

"life\_stage",

when(col("age") < 13, "child")

.when(col("age").between(13, 19), "teenager")

.otherwise("adult"),

)

df1.show()

df1.where(col("life\_stage").isin(["teenager", "adult"])).show()

from pyspark.sql.functions import avg

df1.select(avg("age")).show()

df1.groupBy("life\_stage").avg().show()

spark.sql("select avg(age) from {df1}", df1=df1).show()

spark.sql("select life\_stage, avg(age) from {df1} group by life\_stage", df1=df1).show()

df1.write.saveAsTable("some\_people")

spark.sql("select \* from some\_people").show()

spark.sql("INSERT INTO some\_people VALUES ('frank', 4, 'child')")

spark.sql("select \* from some\_people").show()

spark.sql("select \* from some\_people where life\_stage='teenager'").show()