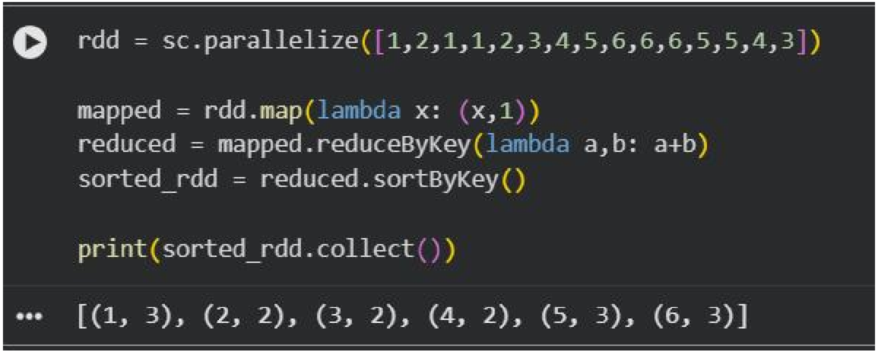
Spark Assignment

Frequency Counting

Problem Statement: Given the list:[1, 2, 1, 1, 2, 3, 4, 5, 6, 6, 6, 5, 5, 4, 3]
Find how many times each number appears and display the result in the form (Number, Count)
Output:[(1,3), (2,2), (3,2), (4,2), (5,3), (6,3)]



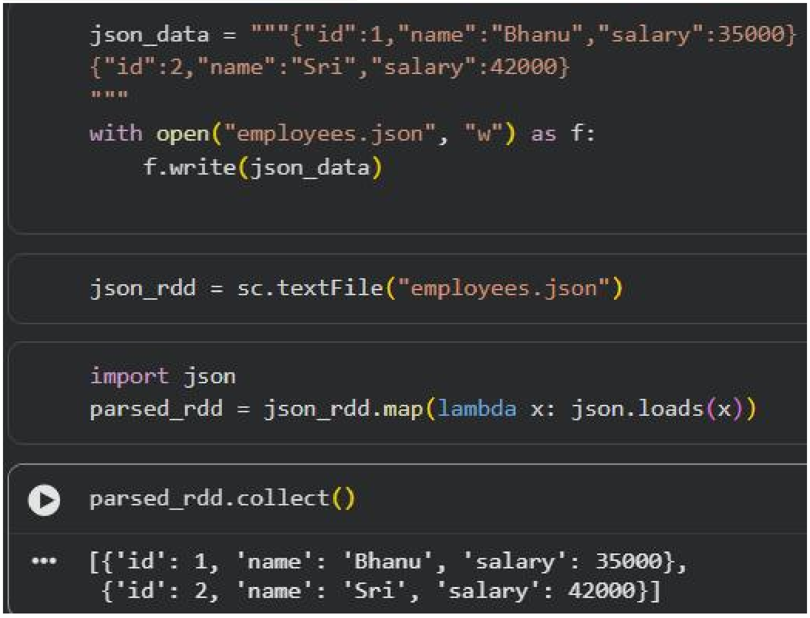
```
rdd = sc.parallelize([1,2,1,1,2,3,4,5,6,6,6,5,5,4,3])

mapped = rdd.map(lambda x: (x,1))
reduced = mapped.reduceByKey(lambda a,b: a+b)
sorted_rdd = reduced.sortByKey()

print(sorted_rdd.collect())
```

```
[(1, 3), (2, 2), (3, 2), (4, 2), (5, 3), (6, 3)]
```

- parallelize() creates an RDD from the list
- map() converts each number into (number,1)
- reduceByKey() adds the counts for same numbers
- sortByKey() sorts the final result

Creating RDD from Different Data Sources

Create RDD from JSON File



```
json_data = """{"id":1,"name":"Bhanu","salary":35000}
{"id":2,"name":"Sri","salary":42000}
"""
with open("employees.json", "w") as f:
    f.write(json_data)
```

```
json_rdd = sc.textFile("employees.json")
```

```
import json
parsed_rdd = json_rdd.map(lambda x: json.loads(x))
```

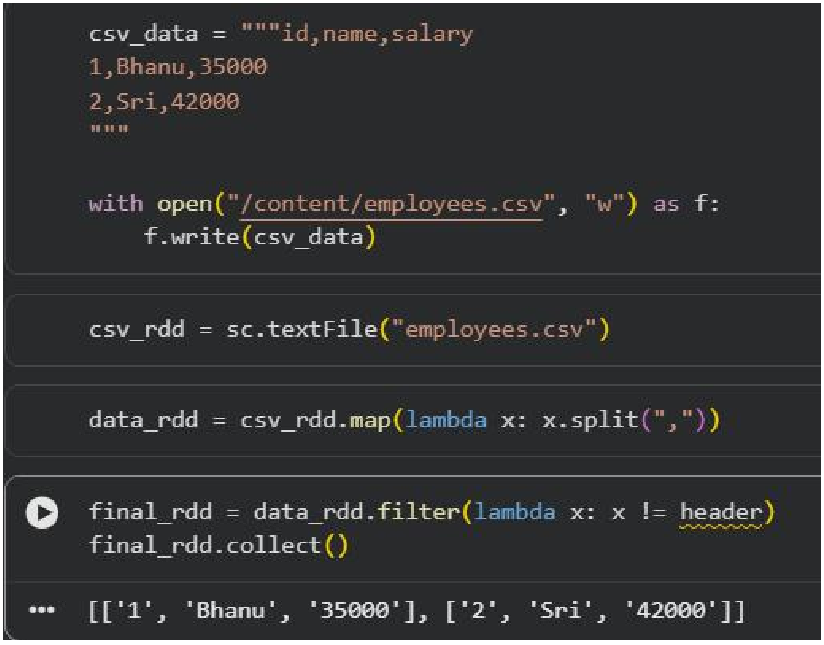
```
parsed_rdd.collect()
```

```
[{'id': 1, 'name': 'Bhanu', 'salary': 35000},
 {'id': 2, 'name': 'Sri', 'salary': 42000}]
```

After creating JSON file
Step 1 – Start SparkContext
Step 2 – Load JSON File as RDD
Step 3 – Convert JSON Text Into Real Data
Step 4 – View the Data

Creating RDD from CSV File

CSV (Comma Separated Values) is a common file format used to store tabular data.



```
csv_data = """id,name,salary
1,Bhanu,35000
2,Sri,42000
"""

with open("/content/employees.csv", "w") as f:
    f.write(csv_data)
```

```
csv_rdd = sc.textFile("employees.csv")
```

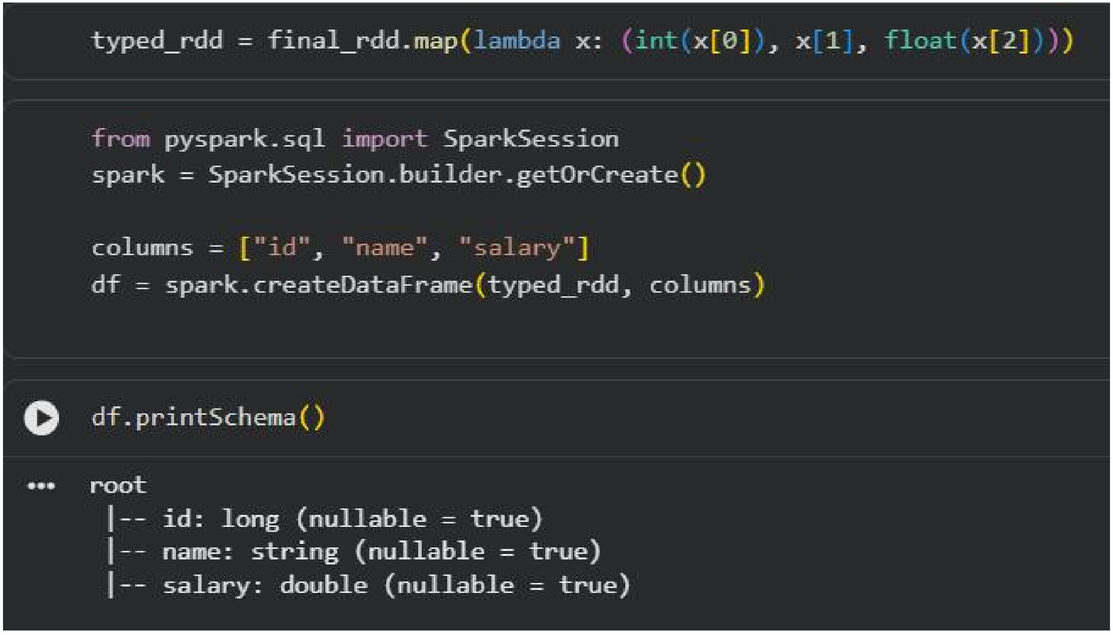
```
data_rdd = csv_rdd.map(lambda x: x.split(","))
```

```
final_rdd = data_rdd.filter(lambda x: x != header)
final_rdd.collect()
```

```
[['1', 'Bhanu', '35000'], ['2', 'Sri', '42000']]
```

How to Create Schema When Creating RDD

A schema means defining what each column represents (name and data type).



```
typed_rdd = final_rdd.map(lambda x: (int(x[0]), x[1], float(x[2])))
```

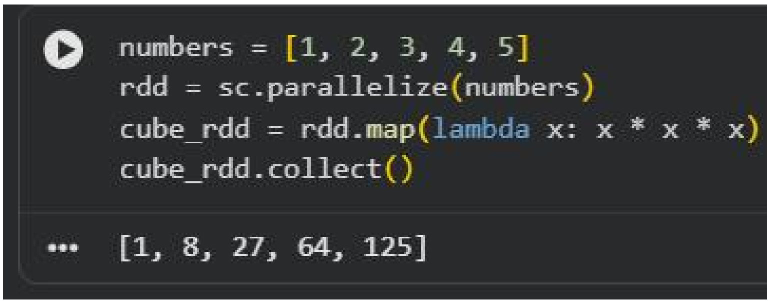
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

columns = ["id", "name", "salary"]
df = spark.createDataFrame(typed_rdd, columns)
```

```
df.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- salary: double (nullable = true)
```

Spark RDD

1) Create an RDD from a list of numbers [1, 2, 3, 4, 5] and cube each number using map().



```
numbers = [1, 2, 3, 4, 5]
rdd = sc.parallelize(numbers)
cube_rdd = rdd.map(lambda x: x * x * x)
cube_rdd.collect()
```

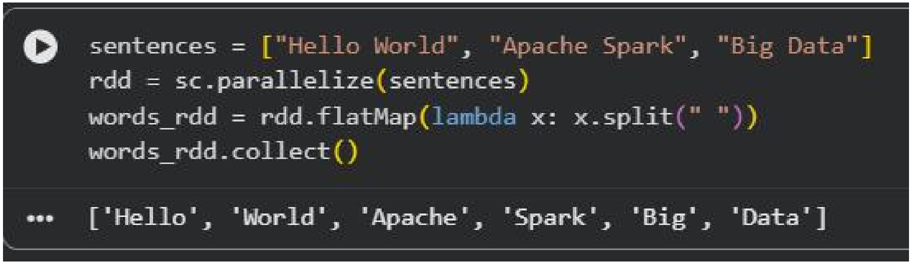
```
[1, 8, 27, 64, 125]
```

parallelize()	Converts normal Python list into RDD
map()	Applies a function to each element
lambda x:x*x*x	Cubes each number
collect()	Brings RDD result back to Python

2) Convert sentences into an RDD of individual words using flatMap().

Data: ["Hello World", "Apache Spark", "Big Data"]

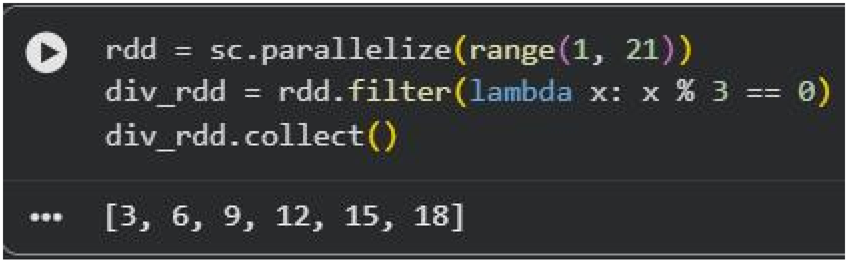
Each sentence has multiple words.We have to split all sentences and create one RDD containing only words.



```
sentences = ["Hello World", "Apache Spark", "Big Data"]
rdd = sc.parallelize(sentences)
words_rdd = rdd.flatMap(lambda x: x.split(" "))
words_rdd.collect()
```

```
['Hello', 'World', 'Apache', 'Spark', 'Big', 'Data']
```

3) Filter an RDD to keep only numbers divisible by 3 from range 1 to 20.



```
rdd = sc.parallelize(range(1, 21))
div_rdd = rdd.filter(lambda x: x % 3 == 0)
div_rdd.collect()
```

```
[3, 6, 9, 12, 15, 18]
```

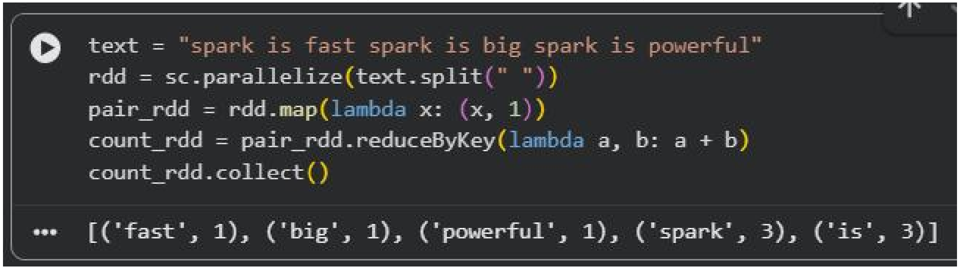
filter()	Keeps only elements that satisfy condition
x % 3 == 0	Checks divisibility by 3

4)Word Count

Implement word count using RDD transformations on the text

"spark is fast spark is big spark is powerful"

We have to count how many times each word appears.



```
text = "spark is fast spark is big spark is powerful"
rdd = sc.parallelize(text.split(" "))
pair_rdd = rdd.map(lambda x: (x, 1))
count_rdd = pair_rdd.reduceByKey(lambda a, b: a + b)
count_rdd.collect()
```

```
[('fast', 1), ('big', 1), ('powerful', 1), ('spark', 3), ('is', 3)]
```

split	Breaks sentence into words
(word,1)	Assigns count 1 to each word
reduceByKey	Adds counts of same words