Ex. No. 7	**Data Partitioning and Repartitioning in Apache Spark**
Youtube Link	**https://www.youtube.com/watch?v=opXIUfkh7HY**
Date of Exercise	3.11.25

AIM

To understand and demonstrate how data partitioning and repartitioning affect performance and data distribution in Apache Spark using RDD operations.

Procedure:

1. Initialize SparkContext.
2. Create an RDD from a list or a file.
3. Check the default number of partitions.
4. Use repartition() to increase or change partitions.
5. Use coalesce() to reduce partitions.
6. Print the number of partitions and elements in each.
7. Observe how repartitioning affects data distribution.

Program:

```
import findspark
findspark.init()

from pyspark import SparkConf, SparkContext

# Initialize SparkContext
conf = SparkConf().setAppName("PartitioningExample").setMaster("local[*]")
sc = SparkContext(conf=conf)

# Sample data
data = list(range(1, 21))  # Numbers from 1 to 20

# Create RDD with 4 partitions
rdd = sc.parallelize(data, 4)

print("Initial number of partitions:", rdd.getNumPartitions())

# Print elements in each partition
def inspect_partitions(index, iterator):
    yield f"Partition {index}: {list(iterator)}"

print("\nOriginal Partition Layout:")
for part in rdd.mapPartitionsWithIndex(inspect_partitions).collect():
    print(part)

# Repartition to 6 partitions (involves full shuffle)
```

```
rdd_repartitioned = rdd.repartition(6)
print("\nAfter repartition(6):")
print("Partitions:", rdd_repartitioned.getNumPartitions())
for part in rdd_repartitioned.mapPartitionsWithIndex(inspect_partitions).collect():
    print(part)

# Coalesce to 2 partitions (no full shuffle)
rdd_coalesced = rdd.coalesce(2)
print("\nAfter coalesce(2):")
print("Partitions:", rdd_coalesced.getNumPartitions())
for part in rdd_coalesced.mapPartitionsWithIndex(inspect_partitions).collect():
    print(part)

# Stop SparkContext
sc.stop()
```

Output:

```
Initial number of partitions: 4
```

```
Original Partition Layout:

Partition 0: [1, 2, 3, 4, 5]

Partition 1: [6, 7, 8, 9]

Partition 2: [10, 11, 12, 13]

Partition 3: [14, 15, 16, 17, 18, 19, 20]


After repartition(6):

Partitions: 6

Partition 0: [..]

Partition 1: [..]

... (elements redistributed across 6 partitions)


After coalesce(2):

Partitions: 2

Partition 0: [..]

Partition 1: [..]
```

Result :

The experiment successfully demonstrated the working of **repartition()** and **coalesce()** are used in Apache Spark to control the number of data partitions.