📋 Lab Overview (30 mins)

Goal: Master RDD core operations hands-on **Tools**: PySpark shell or Databricks notebook **Level**: Beginner → Intermediate

```
Time: 5 + 7 + 7 + 8 + 3 = 30 mins
Covers: RDD creation, transformations, actions, pair RDDs, word
count
```

Section 1: Setup & First RDD

Start PySpark shell:

```
pyspark
```

Copy-paste these:

```
# 1. Create RDD from list
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
rdd = sc.parallelize(nums)

# 2. Basic actions
print("Count:", rdd.count())          # 10
print("First 3:", rdd.take(3))        # [1, 2, 3]
print("Sample:", rdd.takeSample(False, 3))  # Random 3
```

TASK 1.1 (1 min): Create RDD from `["Spark", "RDD", "HandsOn"]` → print count

Section 2: Transformations (7 mins)

Copy-paste & understand:

```
# MAP: Transform each element
squared = rdd.map(lambda x: x*x)
```

```
print("Squared:", squared.collect())  # [1, 4, 9, 16...]

# FILTER: Select subset
evens = rdd.filter(lambda x: x%2 == 0)
print("Evens:", evens.collect())      # [2, 4, 6, 8, 10]

# Chain them
big_squares = rdd.map(lambda x: x*x).filter(lambda x: x > 50)
print("Big squares:", big_squares.collect())
```

TASK 2.1 (2 mins):

```
Create cubes (x*x*x), filter >100, print result
Expected: [125, 216, 343, 512, 729, 1000]
```

TASK 2.2 (1 min):

```
total = rdd.reduce(lambda a,b: a+b)
print("Sum:", total)  # 55
```

Section 3: Text Processing + flatMap

In-memory text data (no file needed):

```
lines = [
    "Apache Spark processes data fast",
    "RDD is resilient distributed dataset",
    "Hands-on practice makes perfect",
    "Transformations are lazy evaluations"
]
text_rdd = sc.parallelize(lines)

print("Lines:", text_rdd.collect())
```

Split into words:

```
# FLATMAP: Split + flatten
words = text_rdd.flatMap(lambda line: line.lower().split())
print("All words:", words.collect())
print("First 5:", words.take(5))
```

TASK 3.1 (2 mins):

```
Clean words: lowercase + remove "data" + length > 3
Print result
```

TASK 3.2 (1 min):

```
Count words with length > 4
```

Section 4: Pair RDDs & Word Count (8 mins)

Complete word count:

```python
# 1. Create (word, 1) pairs
word_pairs = words.map(lambda w: (w, 1))

# 2. Count by word
counts = word_pairs.reduceByKey(lambda a,b: a+b)
print("Word counts:", counts.collect())

# 3. Sort by frequency (descending)
top_words = counts.map(lambda kv: (kv[1], kv[0])) \
                  .sortByKey(ascending=False)
print("Top 3:", top_words.take(3))
```

TASK 4.1 (2 mins):

```
Filter words appearing 2+ times only
Expected: [('is', 1), ('practice', 1), ...] → only frequent ones
```

TASK 4.2 (1 min):

```python
counts.saveAsTextFile("wordcount_output")
```

Section 5: MINI CHALLEN

Transactions data:

```python
transactions = [
    ("Alice", 100), ("Bob", 200), ("Alice", 50),
    ("Charlie", 70), ("Bob", 30), ("Alice", 150)
]
tx_rdd = sc.parallelize(transactions)
```

Challenge (solve in 3 mins):

```
1. Total spend per customer
2. Sort by spend DESCENDING
3. Print top 2 customers
```

Expected output:

```
[('Bob', 230), ('Alice', 300), ('Charlie', 70)]
Top 2: [('Alice', 300), ('Bob', 230)]
```

💡 Pro Tips

```
- Transformations are LAZY (nothing happens till action)
- Use take(10) not collect() for big data
- Local mode = spark.master=local[*]
- Restart: sc.stop(); new SparkContext
```