SCALA + SPARK CHEAT SHEET (Interactive Spark Shell / IntelliJ)

---

1. Start Spark Session

Use this in IntelliJ:

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder()
    .appName("ScalaSparkPractice")
    .master("local[*]")
    .getOrCreate()

import spark.implicits._
```

---

2. RDD BASICS

Create RDD

```
val rdd = spark.sparkContext.parallelize(Seq(1,2,3,4,5))
```

RDD Transformations

```
val squares = rdd.map(x => x * x)
val evens = rdd.filter(_ % 2 == 0)
val pairs = rdd.map(x => (x, x * 10))
```

RDD Actions

```
rdd.collect()
rdd.count()
```

```
rdd.reduce(_ + _)
rdd.first()
```

---

3. DATAFRAME BASICS

Create DataFrame

```
val df = Seq(
  (1, "Ram", 50000),
  (2, "Sam", 60000)
).toDF("id", "name", "salary")

df.show()
df.printSchema()
```

Select & Filter

```
df.select("name").show()
df.filter($"salary" > 55000).show()
df.withColumn("incremented", $"salary" + 1000).show()
```

---

4. COMMON TRANSFORMATIONS

```
df.drop("salary")
df.withColumnRenamed("name","emp_name")
df.sort($"salary".desc)
df.distinct()
```

---

5. AGGREGATIONS

```
import org.apache.spark.sql.functions._
```

```
df.groupBy("name")
  .agg(
    sum("salary").as("total_salary"),
    avg("salary").as("avg_salary")
  )
  .show()
```

---

6. JOINS

```
val dept = Seq(
  (1, "IT"), (2, "HR")
).toDF("id", "dept")

df.join(dept, "id").show()
df.join(dept, Seq("id"), "left").show()
```

---

7. WINDOW FUNCTIONS

```
import org.apache.spark.sql.expressions.Window

val w = Window.orderBy("salary")

df.withColumn("rank", rank().over(w))
  .withColumn("dense_rank", dense_rank().over(w))
  .show()
```

---

8. READ & WRITE FILES

Read Files

```
spark.read.csv("input.csv")
spark.read.json("data.json")
```

```
spark.read.parquet("data.parquet")
```

Write Files

```
df.write.mode("overwrite").json("output/json")
df.write.parquet("output/parquet")
```

---

9. UDF (User Defined Function)

```
val add10 = udf((x: Int) => x + 10)

df.withColumn("new_salary", add10($"salary")).show()
```

---

10. SPARK SQL

Register table

```
df.createOrReplaceTempView("employees")
```

Query

```
spark.sql("SELECT name, salary FROM employees WHERE salary >
55000").show()
```

---

11. DATA CLEANING

```
df.na.fill("Unknown")
df.na.drop()
df.na.replace("salary", Map(50000 -> 55000))
```

---

12. COLUMN FUNCTIONS

```
col("salary")
lit(100)
concat($"name", lit("_emp"))
split($"name", "a")
substr($"name", 1, 2)
```

---

13. CASTING

```
df.withColumn("salary_str", $"salary".cast("string"))
```

---

14. EXPLODE (very important)

```
val df2 = Seq(
  (1, Array("A","B","C"))
).toDF("id","letters")

df2.select($"id", explode($"letters")).show()
```

---

15. CREATE SAMPLE DATA QUICKLY

Employee sample

```
val employees = Seq(
  (1, "Amit", 10000, "IT"),
  (2, "Ravi", 15000, "IT"),
  (3, "Kiran", 12000, "HR"),
  (4, "Neha", 20000, "HR")
).toDF("id","name","salary","dept")
```

---

16. REAL PRACTICE QUESTIONS (WITH CODE)

❓ 1. Find highest salary per dept

```
employees.groupBy("dept")
  .agg(max("salary").as("max_salary"))
  .show()
```

❓ 2. Find 2nd highest salary

```
employees.withColumn("rank",
dense_rank().over(Window.orderBy($"salary".desc)))
  .filter($"rank" === 2)
  .show()
```

❓ 3. Count employees per dept

```
employees.groupBy("dept").count().show()
```

❓ 4. Convert salary to salary + 20%

```
employees.withColumn("new_salary", $"salary" * 1.2).show()
```

---

17. JOIN PRACTICE DATASET

```
val dept = Seq(
  ("IT","Tech"),
  ("HR","Human Resources")
).toDF("dept","description")

employees.join(dept, "dept").show()
```

---

18. DATE FUNCTIONS

```
import org.apache.spark.sql.functions._

val dates = Seq(
  ("2024-01-10")
).toDF("dt")

dates.select(
  current_date(),
  current_timestamp(),
  year($"dt"),
  month($"dt"),
  dayofmonth($"dt"),
  date_add($"dt", 5)
).show()
```

---

19. PERFORMANCE COMMANDS

```
df.explain(true)       // Explain plan
df.cache()             // Cache DF
df.count()             // Trigger cache
df.unpersist()         // Remove from cache
```

---

20. WRITING A COMPLETE SPARK PROGRAM

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object SparkApp {
  def main(args: Array[String]): Unit = {
```

```
    val spark =
SparkSession.builder().master("local[*]").getOrCreate()
    import spark.implicits._

    val df = Seq(
      (1,"Amit",10000),
      (2,"Ravi",20000)
    ).toDF("id","name","salary")

    df.withColumn("tax", $"salary" * 0.10)
      .show()

    spark.stop()
  }
}
```
