Apache Spark 2 using Scala - Basic Transformations using Data Frames

1. Overview — Basic Transformations in Spark DataFrames

In Spark, **DataFrame transformations** are operations that produce a new DataFrame from an existing one **without changing the original data** (immutability).
Think of transformations like **recipes** — you don't eat the raw ingredients directly; you create a new dish from them.

Common Basic DataFrame Transformations

Here are the most used ones in Spark 2 (Scala API):

Transformation	Purpose	Example
select()	Choose specific columns	df.select("name", "age")
withColumn()	Add/replace a column	df.withColumn("age2", col("age") + 2)
drop()	Remove a column	df.drop("age")
filter() / where()	Filter rows	df.filter(col("age") > 25)
distinct()	Remove duplicate rows	df.distinct()
orderBy() / sort()	Sort rows	df.orderBy(col("age").desc)
limit()	Limit rows	df.limit(10)
groupBy()	Group rows for aggregation	df.groupBy("dept").count()
join()	Combine DataFrames	df1.join(df2, "id")

2. Example Code — Spark 2 + Scala

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
object BasicTransformationsExample {
 def main(args: Array[String]): Unit = {
   val spark = SparkSession.builder()
     .appName("BasicTransformationsExample")
     .master("local[*]")
```

```
      .getOrCreate()
    import spark.implicits._
    // Sample data
    val data = Seq(
      (1, "Anjali", 25, "IT"),
      (2, "Benny", 30, "HR"),
      (3, "Chitra", 35, "IT"),
      (4, "Dani", 40, "Finance"),
      (5, "Ella", 30, "IT")
    )
    val df = data.toDF("id", "name", "age", "dept")
    println("=== Original DataFrame ===")
    df.show()
    // 1. Select specific columns
    val selectedDF = df.select("name", "age")
    selectedDF.show()
    // 2. Add a new column
    val newColumnDF = df.withColumn("age_plus_5", col("age") + 5)
    newColumnDF.show()
    // 3. Filter rows
    val filteredDF = df.filter(col("age") > 30)
    filteredDF.show()
    // 4. Remove duplicates
    val distinctDF = df.select("dept").distinct()
    distinctDF.show()
    // 5. Sort data
    val sortedDF = df.orderBy(col("age").desc)
    sortedDF.show()
    // 6. Group and aggregate
    val groupedDF = df.groupBy("dept").agg(avg("age").alias("avg_age"))
    groupedDF.show()
    // 7. Join example
    val deptInfo = Seq(
      ("IT", "Building A"),
      ("HR", "Building B"),
      ("Finance", "Building C")
    ).toDF("dept", "location")
    val joinedDF = df.join(deptInfo, "dept")
    joinedDF.show()
    spark.stop()
  }
}
```

3. Explanation of Each Transformation

- **select()** → Choose specific columns for a smaller dataset.
- **withColumn()** → Creates a new column or updates an existing one with an expression.
- **filter() / where()** → Keeps only rows meeting a condition.
- **distinct()** → Removes duplicate rows.
- **orderBy() / sort()** → Orders data by one or more columns.
- **groupBy()** → Groups rows for aggregations (sum, avg, count, etc.).
- **join()** → Combines rows from two DataFrames based on a condition.

4. Recommended GitHub Repositories

Here are some **good repositories** where you can explore **Spark 2 + Scala examples**:

1. spark-scala-examples by sparkbyexamples
2. awesome-spark
3. spark-tutorials by datamechanics
4. Spark-Scala-Cookbook

5. Quick Quiz — Test Your Understanding

1. What is the difference between select() and withColumn()?
2. How do you filter rows where age is greater than 40?
3. Which transformation removes duplicate rows in a DataFrame?
4. How do you sort rows in descending order of salary?
5. What does the following code do?

```
df.groupBy("dept").agg(count("*").alias("total_employees"))
```