Apache Spark 2 using SQL - Basic Transformations

1. Concept Overview

In Spark SQL, **basic transformations** refer to operations that **transform one DataFrame/Dataset into another** without changing the underlying data source directly.
These transformations are **lazy** (executed only when an action is called) and can include:

- Column selection & renaming
- Filtering rows
- Adding new columns
- Dropping columns
- Changing data types
- Aggregations (groupBy)
- Ordering and limiting results

Key points:

- Spark SQL transformations are similar to SQL queries but work in a distributed manner.
- Can be written in **SQL syntax** or **DataFrame API** in Scala.

2. Example Dataset

Let's assume we have an employees.csv:

emp_id	name	department	salary	age
1	John	Sales	45000	30
2	Abe	IT	60000	28
3	Bob	HR	35000	35
4	Charlie	IT	70000	40

3. Scala Code – Basic Transformations Using SQL

```
import org.apache.spark.sql.{SparkSession, functions => F}
object SparkSQLBasicTransformations {
```

```
  def main(args: Array[String]): Unit = {
    // 1. Spark Session
    val spark = SparkSession.builder()
      .appName("Spark SQL Basic Transformations")
      .master("local[*]")
      .getOrCreate()
    import spark.implicits._
    // 2. Read CSV into DataFrame
    val df = spark.read.option("header", "true").option("inferSchema", "true")
      .csv("employees.csv")
    // Register as SQL temporary view
    df.createOrReplaceTempView("employees")
    println("=== Original Data ===")
    df.show()
    // 3. SELECT specific columns
    val selectDF = spark.sql("SELECT name, salary FROM employees")
    println("=== Selected Columns ===")
    selectDF.show()
    // 4. Filter rows (salary > 50,000)
    val filterDF = spark.sql("SELECT * FROM employees WHERE salary > 50000")
    println("=== Filtered Rows ===")
    filterDF.show()
    // 5. Add new column (bonus = 10% of salary)
    val bonusDF = spark.sql("SELECT *, salary * 0.10 AS bonus FROM employees")
    println("=== Added Bonus Column ===")
    bonusDF.show()
    // 6. Rename column
    val renameDF = df.withColumnRenamed("department", "dept")
    println("=== Renamed Column ===")
    renameDF.show()
    // 7. Group by and aggregation
    val groupDF = spark.sql(
      "SELECT department, AVG(salary) AS avg_salary FROM employees GROUP BY department"
    )
    println("=== Average Salary by Department ===")
    groupDF.show()
    // 8. Order by
    val orderDF = spark.sql("SELECT * FROM employees ORDER BY salary DESC")
    println("=== Ordered by Salary Desc ===")
    orderDF.show()
    spark.stop()
  }
}
```

4. Explanation of Transformations

Transformation	Purpose	Example in SQL	Example in DataFrame API
Select	Pick specific columns	SELECT name, salary FROM employees	df.select("name", "salary")
Filter	Keep rows meeting condition	WHERE salary > 50000	df.filter($"salary" > 50000)

Add Column	Create derived column	salary * 0.10 AS bonus	df.withColumn("bonus", $"salary" * 0.10)
Rename	Change column name	N/A in SQL alias	df.withColumnRenamed("department", "dept")
Group & Aggregate	Summary stats	GROUP BY department	df.groupBy("department").avg("salary")
Order By	Sort data	ORDER BY salary DESC	df.orderBy($"salary".desc)

5. Example Run Output

```
=== Selected Columns ===
+-------+------+
| name  |salary|
+-------+------+
| John  |45000 |
| Abe |60000 |
| Bob   |35000 |
| Charlie|70000|
+-------+------+
=== Filtered Rows ===
+------+-------+-----------+------+---+
|emp_id|name   |department |salary|age|
+------+-------+-----------+------+---+
|2     |Abe  |IT         |60000 |28 |
|4     |Charlie|IT         |70000 |40 |
+------+-------+-----------+------+---+
```

6. Quizzes

Q1: What is the difference between a transformation and an action in Spark SQL?
Q2: Which method registers a DataFrame as a SQL table in Spark?
Q3: Write a Spark SQL query to find employees whose age is greater than 30.
Q4: What does withColumn do in Spark DataFrame API?
Q5: Is ORDER BY in Spark SQL guaranteed to return globally sorted results?

7. GitHub Repositories to Explore

- Spark-Scala-Examples – Covers SQL, DataFrames, RDDs, and transformations.
- awesome-spark – Curated list of Spark resources.
- Spark SQL Examples – SQL-focused Spark transformations.
- apache/spark – Official Spark repository with examples.