Basic Structured Opereations

Keywork	Explanation	Example
schema	display schema in a scala way	df.schema
print schema	display schema structured	df.printSchema()
manual schema	manual schema	import org.apache.spark.sql.types._ val myManualSchema = StructType(Array(StructField("DEST_COUNTRY_NAME", StringType, true), StructField("ORIGIN_COUNTRY_NAME", StringType, true), StructField("count", LongType, false, Metadata.fromJson("{\"hello\":\"world\"}")))) val df = spark.read.format("json").schema(myManualSchema) .load("/FileStore/tables/2015_summary-ebaee.json")
select	select a col	import org.apache.spark.sql.functions._ df.select("count").show(3)
columns	list cols	df.columns
select + expr	add 5 to all specific col values	df.select(expr("colName + 5")).show(3)
row	create a row / record	import org.apache.spark.sql.Row val myRow = Row("Hello", null, 1, false)
() getString / Int	get elt of a row by index, type with idx	myRow(0) myRow.getString(0) myRow.getInt(2)
	create a df from rows	val myManualSchema = new StructType(Array(new StructField("some", StringType, true), new StructField("col", StringType, true), new StructField("names", LongType, false))) val myRows = Seq(Row("Hello", null, 1L)) val myRDD = spark.sparkContext.parallelize(myRows) val myDf = spark.createDataFrame(myRDD, myManualSchema)
as .alias	change col name	df.select(expr("colName AS newColName")).show(2) df.select(expr("colName").alias("newColName")).show(2) spark.sql("colName AS newColName FROM dfTable LIMIT 2").show()
	create new col made of comparison between 2 cols	df.selectExpr("*", "(DEST_COUNTRY_NAME = ORIGIN_COUNTRY_NAME) as whitinCountry") same thing with SQL but without quotes
avg / count(distinct)	avg & count distinct	df.selectExpr("avg(colNameA)", "count(distinct(colNameB))")
lit as	Converting to Spark Types (Literals)	df.select(expr("*"), lit(1).as("One")).show(2)
withColumn	adding col	df.withColumn("numberOne", lit("ones"))
withColumn ==	adding col with condition	df.withColumn("cond", expr("colNameA == colNameB"))
withColumn	duplicate col with new name	df.withColumn("newName", expr("oldName"))
drop	drop columns	df.drop("colNameA", "colNameB"
cast	Changing a Column’s Type (cast)	df.withColumn("count2", col("count").cast("long"))
filter where	Filtering Rows - numerical condition	df.filter(col("count") < 2) df.where("count < 2") spark.sql("SELECT * FROM dfTable WHERE count < 2 LIMIT 2")
where	2 Filters - numerical & text condition	df.where(col("colNameB") < 2) .where(col("colName") =!= "someText").show(2) spark.sql(""" SELECT * FROM dfTable WHERE colNameB < 2 AND colName != "someText" LIMIT 2""")
distinct().count()	Getting Unique Rows	df.select("colNameA", "colNameB").distinct().count()
distinct().count()	Getting Unique values of a col	df.select("colNameA").distinct().count()
sample	Random Samples	val seed = 5 val withReplacement = false val fraction = 0.5 df.sample(withReplacement, fraction, seed)
randomSplit	Random Splits	val dataFrames = df.randomSplit(Array(0.25, 0.75), seed)
sort orderBy	Sorting Rows	df.sort(desc("colNameA")).show(5) df.orderBy("colNameA", "colNameB").show(5) df.orderBy(expr("colNameA desc")).show(2) df.orderBy(desc("colNameA"), asc("colNameB")).show(2)
getNumPartitions	Get Num of partitions	df.rdd.getNumPartitions
repartition(nb)	Repartition	val tempDf = df.repartition(5) df.repartition(5, col("colNameA"))
coalesce(nb)		df.repartition(5, col("colNameA")).coalesce(2)
.toLocalIterator()	to local iterator	val collectDF = df.limit(10) collectDF.take(5) collectDF.toLocalIterator()