# SQL to PySpark Basic Command Conversion Cheatsheet

(tables for SQL are table, left_table and right_table, dataframes for PySpark are df, left_df and right_df. Columns are col_1, col_2, col_3, col_4 in both cases)

| Action | SQL<br>*Assuming tables have been created using*<br>***df.createOrReplaceTempView("table")*** | PySpark<br>*Assuming* ***import pyspark.sql.functions as fn*** |
|---|---|---|
| Describe | df_sql = spark.sql("DESCRIBE table")<br>df_sql.show() | df.printSchema() |
| | | |
| Selection and aliasing | spark.sql("SELECT col_1 AS f1, col_3 AS f3 FROM table") | df.select(fn.col("col_1").alias("f1"), fn.col("col_3").alias("f3")) |
| Select distinct | spark.sql("SELECT DISTINCT col_4 FROM table") | df.select(fn.col("col_4")).distinct() |
| Limit results | spark.sql("SELECT * FROM table LIMIT 2") | df.limit(2) |
| Ascending order | spark.sql("SELECT * FROM table ORDER BY col_2") | df.orderBy("col_2") |
| Descending order | spark.sql("SELECT * FROM table ORDER BY col_1 DESC") | df.orderBy("col_1", ascending = False) |
| | | |
| Filter | spark.sql("SELECT * FROM table WHERE col_1 > 3") | df.filter(fn.col("col_1") > fn.lit(3)) |
| Group by and agggregation | spark.sql("SELECT col_4, COUNT(col_1), SUM (col_2) FROM table GROUP BY col_4") | df.groupBy("col_4").agg(fn.count("col_1"), fn.sum("col_2")) |
| | | |
| Inner join | spark.sql("SELECT * FROM left_table INNER JOIN right_table ON left_table.left_1 = right_table.right_1") | left_df.join(right_df, left_df.left_1 == right_df.right_1) |
| Outer join | spark.sql("SELECT * FROM left_table FULL OUTER JOIN right_table ON left_1 = right_1") | left_df.join(right_df, left_df.left_1 == right_df.right_1, how = "outer") |
| Left join | spark.sql("SELECT * FROM left_table LEFT JOIN right_table ON left_1 = right_1") | left_df.join(right_df, left_df.left_1 == right_df.right_1, how = "left") |
| Cross Join | spark.sql("SELECT * FROM left_table CROSS JOIN right_table") | left_df.crossJoin(right_df) |
| | | |
| Union | spark.sql("SELECT left_1 FROM left_table UNION SELECT Right_1 FROM right_table") | left_df.select("left_1").union(right_df.select("right_1")).distinct() |
| Union all | spark.sql("SELECT left_1 FROM left_table UNION ALL SELECT Right_1 FROM right_table") | left_df.select("left_1").union(right_df.select("right_1")) |
| | | |
| Amending column (Not SQL ALTER) | spark.sql("SELECT col_1, col_2, col_3, CASE WHEN col_4 = 'two' THEN 'changed' ELSE col_4 END AS col_4 FROM table") | df.withColumn("col_4", fn.when(df.col_4 == 'two', 'changed').otherwise(df.col_4)) |

compiled for my blog at www.justinmatters.co.uk/wp/ for work book see