PySpark SQL CHEAT SHEET

Initializing Spark Session

- `>>> from pyspark.sql import SparkSession`
- `>>> spark = SparkSession\.builder\.appName("PySpark SQL\.config("spark.some.config.option", "some-value") \.getOrCreate()`

Creating Data Frames

```
#import pyspark class Row from module sql
        >>>from pyspark.sql import *
```

- Infer Schema:

```
        >>> sc = spark.sparkContext
        >>> A = sc.textFile("Filename.txt")
        >>> B = lines.map(lambda x: x.split(","))
        >>> C = parts.map(lambda a: Row(col1=a[0],col2=int(a[1])))
        >>> C_df = spark.createDataFrame(C)
```

- Specify Schema:

```
        >>> C = parts.map(lambda a: Row(col1=a[0], col2=int(a[1].strip())))
        >>> schemaString = "MyTable"
        >>> D = [StructField(field_name, StringType(), True) for
        field_name in schemaString.split()]
        >>> E = StructType(D)
        >>> spark.createDataFrame(C, E).show()
```

col1	col2
row1	3
row2	4
row3	5

FURTHERMORE: Spark, Scala and Python Training Training Course

From Spark Data Sources

- JSON

```
        >>>df = spark.read.json("table.json")
        >>>df.show()
        >>> df2 = spark.read.load("tablee2.json", format="json")
```

- Parquet Files

```
        >>> df3 = spark.read.load("newFile.parquet")
```

Inspect Data

```
• >>> df.dtypes              -- Return df column names and data types
• >>> df.show()              -- Display the content of df
• >>> df.head()              -- Return first n rows
• >>> df.first(n)            -- Return the first n rows
• >>> df.schema              -- Return the schema of df
• >>> df.describe().show()   -- Compute summary statistics
• >>> df.columns             -- Return the columns of df
• >>> df.count()             -- Count the number of rows in df
• >>> df.distinct().count()  -- Count the number of distinct rows in df
• >>> df.printSchema()       -- Print the schema of df
• >>> df.explain()           -- Print the (logical and physical) plans
```

Column Operations

- Add

```
>>> df = df.withColumn('col1',df.table.col1) \ .withColumn('col2',df.table.col2) \
.withColumn('col3',df.table.col3) \ .withColumn('col4',df.table.col4)
\.withColumn(col5', explode(df.table.col5))
```

- Update

```
>>> df = df.withColumnRenamed('col1', 'column1')
```

- Remove

```
>>> df = df.drop("col3", "col4")
>>> df = df.drop(df.col3).drop(df.col4)
```

Actions

- Group By: `>>> df.groupBy("col1")\ .count() \ .show()`
- Filter: `>>> df.filter(df["col2"]>4).show()`
- Sort:

```
        >>> peopledf.sort(peopledf.age.desc()).collect()
        >>> df.sort("col1", ascending=False).collect()
        >>> df.orderBy(["col1","col3"],ascending=[0,1])\ .collect()
```

- Missing & Replacing Values:

```
        >>> df.na.fill(20).show()
        >>> df.na.drop().show()
        >>> df.na \ .replace(10, 20) \ .show()
```

- Repartitioning:

```
        >>> df.repartition(10)\ df with 10 partitions .rdd \
.getNumPartitions()
        >>> df.coalesce(1).rdd.getNumPartitions()
```

SQL Queries

```
>>> from pyspark.sql import functions as f
```

- Select

```
>>> df.select("col1").show()
>>> df.select("col2","col3") \ .show()
```

- When

```
>>> df.select("col1", f.when(df.col2> 30, 1) \ .otherwise(0)) \ .show()
>>> df[df.col1.isin("A","B")] .collect()
```

Running SQL Queries Programmatically

- Registering Data Frames as Views:

```
>>> peopledf.createGlobalTempView("column1")
>>> df.createTempView("column1")
>>> df.createOrReplaceTempView("column2")
```

- Query Views

```
>>> df_one = spark.sql("SELECT * FROM customer").show()
>>> df_new = spark.sql("SELECT * FROM global_temp.people")\ .show()
```

Output Operations

- Data Structures:

```
>>> rdd_1 = df.rdd
>>> df.toJSON().first()
>>> df.toPandas()
```

- Write & Save to Files:

```
>>> df.select("Col1", "Col2")\ .write \ .save("newFile.parquet")
>>> df.select("col3", "col5") \ .write \ .save("table_new.json",format="json")
```

- Stopping SparkSession

```
>>> spark.stop()
```