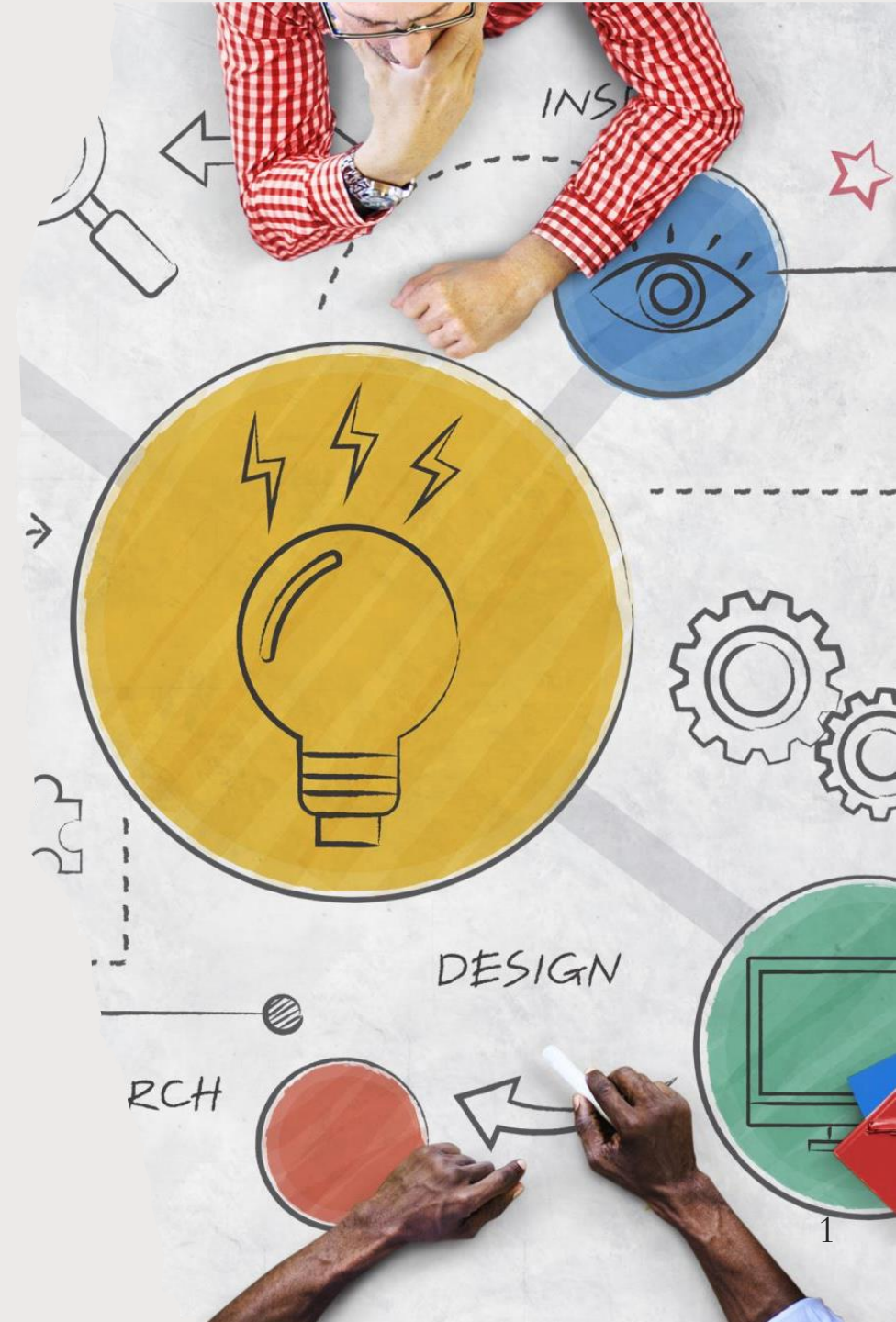


ACTIONS & BASIC TRANSFORMATIONS



CONTENT

Introduction & Presentation ETL Process

Actions & Basic Transformations

Advanced Transformations

Databricks

GOAL OF THE LEARNING SECTIONS



- Perform Basic Transformation
- Perform Conditional Selection of Rows
- Carry out basic Data Cleaning

IMPORTING DATA

- Importing Data with `.spark.read.options()`

```
df1 = spark.read.options(header='True',  
                           multiline='True',  
                           inferSchema='True')  
      .csv('iris.csv')
```

```
df2 = spark.read.option("multiline",True).json('iris.json')
```

ACCESSING COLUMNS

- Accessing columns with `.select()`

petalLength	petalWidth	sepalLength	sepalWidth	species
1.4	0.2	5.1	3.5	setosa
1.4	0.2	4.9	3.0	setosa
1.3	0.2	4.7	3.2	setosa
1.5	0.2	4.6	3.1	setosa
1.4	0.2	5.0	3.6	setosa
1.7	0.4	5.4	3.9	setosa
1.4	0.3	4.6	3.4	setosa
1.5	0.2	5.0	3.4	setosa
1.4	0.2	4.4	2.9	setosa
1.5	0.1	4.9	3.1	setosa

```
[ ] df1.select("petalLength").show(5)
```

```
+-----+  
|petalLength|  
+-----+  
|         1.4|  
|         1.4|  
|         1.3|  
|         1.5|  
|         1.4|  
+-----+
```

ACCESSING ROWS

- Accessing columns with workaround `.collect()` then access with `print()`

(and square brackets-operator for row)

petalLength	petalWidth	sepalLength	sepalWidth	species
1.4	0.2	5.1	3.5	setosa
1.4	0.2	4.9	3.0	setosa
1.3	0.2	4.7	3.2	setosa
1.5	0.2	4.6	3.1	setosa
1.4	0.2	5.0	3.6	setosa
1.7	0.4	5.4	3.9	setosa
1.4	0.3	4.6	3.4	setosa
1.5	0.2	5.0	3.4	setosa
1.4	0.2	4.4	2.9	setosa
1.5	0.1	4.9	3.1	setosa



```
[ ] # Returns list of Row objects
local_df1 = df1.collect()
print(f"Type of entries: {type(local_df1[0])}\n")
print(f"Entries: {local_df1[:5]}")
```

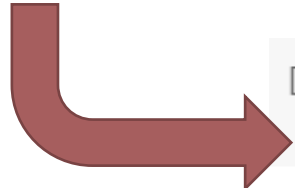
Type of entries: <class 'pyspark.sql.types.Row'>

Entries: [Row(petalLength=1.4, petalWidth=0.2, sepalLength=5.1, sepalWidth=3.5, species='setosa'), Row(petalLength=1.4, petalWidth=0.2, sepalLength=4.9, sepalWidth=3.0, species='setosa'), Row(petalLength=1.3, petalWidth=0.2, sepalLength=4.7, sepalWidth=3.2, species='setosa'), Row(petalLength=1.5, petalWidth=0.2, sepalLength=4.6, sepalWidth=3.1, species='setosa'), Row(petalLength=1.4, petalWidth=0.2, sepalLength=5.0, sepalWidth=3.6, species='setosa')]

ADDING COLUMNS

- Adding columns with `.withColumn()`

petalLength	petalWidth	sepalLength	sepalWidth	species
1.4	0.2	5.1	3.5	setosa
1.4	0.2	4.9	3.0	setosa
1.3	0.2	4.7	3.2	setosa
1.5	0.2	4.6	3.1	setosa
1.4	0.2	5.0	3.6	setosa



```
[ ] df_extraCol = df1.withColumn('newColumn', df1.petalWidth + df1.petalLength)
df_extraCol.show(5)
```

petalLength	petalWidth	sepalLength	sepalWidth	species	newColumn
1.4	0.2	5.1	3.5	setosa	1.5999999999999999
1.4	0.2	4.9	3.0	setosa	1.5999999999999999
1.3	0.2	4.7	3.2	setosa	1.5
1.5	0.2	4.6	3.1	setosa	1.7
1.4	0.2	5.0	3.6	setosa	1.5999999999999999

REMOVING COLUMNS

- Removing columns with `.drop()`

petalLength	petalWidth	sepalLength	sepalWidth	species	petalSum
1.4	0.2	5.1	3.5	setosa	1.5999999999999999
1.4	0.2	4.9	3.0	setosa	1.5999999999999999
1.3	0.2	4.7	3.2	setosa	1.5
1.5	0.2	4.6	3.1	setosa	1.7
1.4	0.2	5.0	3.6	setosa	1.5999999999999999



```
[ ] df1 = df_extraCol.drop(df_extraCol.petalSum)
df1.show(5)
```

petalLength	petalWidth	sepalLength	sepalWidth	species
1.4	0.2	5.1	3.5	setosa
1.4	0.2	4.9	3.0	setosa
1.3	0.2	4.7	3.2	setosa
1.5	0.2	4.6	3.1	setosa
1.4	0.2	5.0	3.6	setosa

CONCATENATING DATAFRAMES

- Concatenating dataframes with `.union()`

petalLength	petalWidth	sepalLength	sepalWidth	species
null	null	null	null	null
1.4	0.2	5.1	3.5	setosa
1.4	0.2	4.9	3.0	setosa
4.1	1.3	5.7	2.8	versicolor
4.1	1.3	5.7	2.8	versicolor
6.0	2.5	6.3	3.3	virginica
5.1	1.9	5.8	2.7	virginica
null	null	null	null	null

petalLength	petalWidth	sepalLength	sepalWidth	species
5.1	1.8	5.9	3.0	virginica

```
df1.union(df2).show()
```

petalLength	petalWidth	sepalLength	sepalWidth	species
null	null	null	null	null
1.4	0.2	5.1	3.5	setosa
1.4	0.2	4.9	3.0	setosa
4.1	1.3	5.7	2.8	versicolor
4.1	1.3	5.7	2.8	versicolor
6.0	2.5	6.3	3.3	virginica
5.1	1.9	5.8	2.7	virginica
null	null	null	null	null
5.1	1.8	5.9	3.0	virginica

PERFORM CONDITIONAL SELECTION OF ROWS

- Combining aggregate functions, to achieve exact selection: `.filter()`

`.where()`

`.withColumn()`

`...`

petalLength	petalWidth	sepalLength	sepalWidth	species
4.2	1.2	5.7	3.0	versicolor
4.2	1.3	5.7	2.9	versicolor
4.3	1.3	6.2	2.9	versicolor
3.0	1.1	5.1	2.5	versicolor
4.1	1.3	5.7	2.8	versicolor
6.0	2.5	6.3	3.3	virginica
5.1	1.9	5.8	2.7	virginica
5.9	2.1	7.1	3.0	virginica
5.6	1.8	6.3	2.9	virginica
5.8	2.2	6.5	3.0	virginica

```
[ ] (df1.select("species", "petalWidth")
      .filter(df1.species == "virginica")
      .dropDuplicates()
      .show(5))
```

species	petalWidth
virginica	2.2
virginica	1.7
virginica	2.5
virginica	2.4
virginica	1.6

PERFORM CONDITIONAL SELECTION OF ROWS



```
[ ] (df1.select("species", "petalWidth", "petalLength")
     .where((df1.species == "setosa") & (df1.petalLength > 1.3))
     .withColumn("petalSum", df1.petalWidth + df1.petalLength)
     .dropDuplicates()
     .describe()
     .show(5))
```

	summary	species	petalWidth	petalLength	petalSum
	count	16	16	16	16
	mean	null	0.30000000000000004	1.5999999999999999	1.9000000000000001
	stddev	null	0.1414213562373095	0.15916448515084428	0.24221202832779934
	min	setosa	0.1	1.4	1.5
	max	setosa	0.6	1.9	2.3

BASIC DATA CLEANING

- Removing NAs with `.dropna()`
- Removing duplicates with `.dropDuplicates()`



```
[ ] (df1.select("petalWidth")  
    .dropna()  
    .dropDuplicates()  
    .show(n=5))
```

```
+-----+  
|petalWidth|  
+-----+  
|      2.4|  
|      0.2|  
|      1.4|  
|      1.7|  
|      2.3|  
+-----+
```