

ACTIONS & BASIC TRANSFORMATIONS



CONTENT

Introduction & Presentation ETL Process

Actions & Basic Transformations

Advanced Transformations

Datavault

GOAL OF THE LEARNING SECTIONS



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

IMPORTING DATA

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

```
df_fru = spark.read.options(header='True',  
                             multiline='True',  
                             inferSchema='True')  
                             .csv('Fruits.csv')
```

```
df_veg = spark.read.option("multiline",True).json('Vegetables.json')
```

ACCESSING COLUMNS

- Accessing columns with `.select()`

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	4	14	0	0
strawberries	7	5	14	0	5
strawberries	7	6	18	30	10
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	7	20	60	25




```
df_fru.select('week').show(5)
```

```
+----+  
|week|  
+----+  
|  1 |  
|  2 |  
|  3 |  
|  4 |  
|  5 |  
+----+
```

ACCESSING ROWS

- Accessing columns with workaround `.collect()` then access with `print()`
(and square brackets-operator for row)

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	4	14	0	0
strawberries	7	5	14	0	5
strawberries	7	6	18	30	10
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	7	20	60	25



```
df_fru_lokal = df_fru.collect()
print(f"Type of entries: {type(df_fru_lokal[0])}\n")
print(f"Entries: {df_fru_lokal[2]}")
```

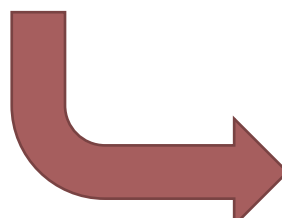
Type of entries: <class 'pyspark.sql.types.Row'>

Entries: Row(crop='strawberries', field=7, week=3, water_consumption=12, revenue=0, yield_per_sqm=0)

ADDING COLUMNS

- Adding columns with `.withColumn()`

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	7	20	60	25
strawberries	7	8	26	150	25
strawberries	7	9	24	150	25
strawberries	7	10	10	100	25
strawberries	7	11	null	150	0



```
df_extraCol = df_fru.withColumn('newColumn', df_fru.yield_per_sqm * df_fru.revenue)
df_extraCol.show()
```

crop	field	week	water_consumption	revenue	yield_per_sqm	newColumn
strawberries	7	7	20	60	25	1500
strawberries	7	8	26	150	25	3750
strawberries	7	9	24	150	25	3750
strawberries	7	10	10	100	25	2500
strawberries	7	11	null	150	0	0

REMOVING COLUMNS

- Removing columns with `.drop()`

crop	field	week	water_consumption	revenue	yield_per_sqm	newColumn
strawberries	7	1	12	0	0	0
strawberries	7	2	10	0	0	0
strawberries	7	3	12	0	0	0
strawberries	7	4	14	0	0	0
strawberries	7	5	14	0	5	0



```
df_fru_2 = df_extraCol.drop(df_extraCol.newColumn)
df_fru_2.show(5)
```

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	4	14	0	0
strawberries	7	5	14	0	5

BASIC DATA CLEANING

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

week	crop	water_consumption	field	revenue	yield_per_sqm
1	strawberries	12	7	0	0
1	strawberries	12	7	0	0
2	strawberries	10	7	0	0
2	strawberries	10	7	0	0
3	strawberries	12	7	0	0
3	strawberries	12	7	0	0
4	strawberries	14	7	0	0
5	strawberries	14	7	0	5
6	strawberries	18	7	30	10
7	strawberries	20	7	60	25
8	strawberries	26	7	150	25
9	strawberries	24	7	150	25
10	strawberries	10	7	100	25
11	strawberries	null	7	150	0



```
(df_fru.select('water_consumption')  
        .dropna()  
        .dropDuplicates()  
        .show())
```

water_consumption
26
12
20
10
24
14
18

CONCATENATING DATAFRAMES

- Concatenating dataframes with `.union()`

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	4	14	0	0
strawberries	7	5	14	0	5

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	6	18	30	10

`df_fru1.union(df_fru2).show()`

crop	field	week	water_consumption	revenue	yield_per_sqm
strawberries	7	1	12	0	0
strawberries	7	2	10	0	0
strawberries	7	3	12	0	0
strawberries	7	4	14	0	0
strawberries	7	5	14	0	5
strawberries	7	6	18	30	10

PERFORM CONDITIONAL SELECTION OF ROWS

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

`.where()`

`.withColumn()`

`...`

week	crop	water_consumption	field	revenue	yield_per_sqm
1	strawberries	12	7	0	0
1	strawberries	12	7	0	0
2	strawberries	10	7	0	0
2	strawberries	10	7	0	0
3	strawberries	12	7	0	0
3	strawberries	12	7	0	0
4	strawberries	14	7	0	0
5	strawberries	14	7	0	5
6	strawberries	18	7	30	10
7	strawberries	20	7	60	25
8	strawberries	26	7	150	25
9	strawberries	24	7	150	25
10	strawberries	10	7	100	25
11	strawberries	null	7	150	0

```
(df_fru.select('week', 'water_consumption', 'revenue')  
  .filter(df_fru.week > 5)  
  .dropna()  
  .show())
```

week	water_consumption	revenue
6	18	30
7	20	60
8	26	150
9	24	150
10	10	100

PERFORM CONDITIONAL SELECTION OF ROWS



```
(df_fru.select('*')
        .where((df_fru.revenue != 0) & (df_fru.week > 7))
        .withColumn('lucrativeness', df_fru.revenue * df_fru.yield_per_sqm)
        .dropna()
        .describe()
        .show())
```

summary	crop	field	week	water_consumption	revenue	yield_per_sqm	lucrative ^{ness}
count	3	3	3	3	3	3	3
mean	null	7.0	9.0	20.0	133.33333333333334	25.0	3333.3333333333335
stddev	null	0.0	1.0	8.717797887081348	28.867513459481287	0.0	721.6878364870322
min	strawberries	7	8	10	100	25	2500
max	strawberries	7	10	26	150	25	3750

FUNCTIONS AND ADVANCED TRANSFORMATIONS



CONTENT

Introduction & Presentation ETL Process

Actions & Basic Transformations

Advanced Transformations

DataVault

GOAL OF THE LEARNING SECTIONS



- How to use powerful Methods like groupBy, Join and Spark functions
- How use Alter data based by using Lambda

INTRO TO SPARK FUNCTIONS ?

- In general, it is possible to use functions from other libraries, such as numpy, on Spark DataFrame objects
- However, this runs counter to the purpose of Spark.
- Focus on optimising the performance of transformation pipelines

Support for Jeremy:

- Using these functions, we can help Jeremy map all records in the maize whose value is above, say, 0, to 10 and all others to 50
- Although one would not initially think of such an if-else statement as a function in Python, it is implemented this way in PySpark by the function `.when()`

INTRO TO SPARK FUNCTIONS ?

Command

```
(df_corn.select("crop",  
  func.when(df_corn["revenue"] < 30, 10)  
  .otherwise(50).alias("revenue under 30"))  
  .show(10))
```

Dataset

crop	field	revenue	water_consumption	week	yield_per_sqm
barley	8	0	12	1	0
barley	8	0	10	2	0
barley	8	0	12	3	0
barley	8	35	14	4	10
barley	8	50	14	5	25
barley	8	40	18	6	15
barley	8	60	12	7	30
corn	10	0	10	1	0
corn	10	0	12	2	0
corn	10	30	20	3	25



Result

crop	revenue under 30
barley	10
barley	10
barley	10
barley	50
barley	50
barley	50
barley	50
barley	50
corn	10
corn	10
corn	50

USER DEFINED FUNCTIONS

Command

```
def firstUpper(s: str) -> str:
    s = s[0].upper() + s[1:]
    return s

firstUpper_UDF = func.udf(firstUpper, "STRING")
# Apply function to our DataFrame containing the Iris data.
df_corn.select(firstUpper_UDF("crop").alias("Crop")).show(10)
```



Result

```
+-----+
| Crop |
+-----+
| Barley |
| Barley |
| Barley |
| Barley |
| Barley |
| Barley |
| Barley |
| Corn |
| Corn |
| Corn |
+-----+
```

- Pass function to Spark Session using `spark.udf.register()`.
- After passing the function it can then be used in all Spark pipelines
- Self-defined functions can be used in the same way as the functions from the Functions module.

USER DEFINED FUNCTIONS

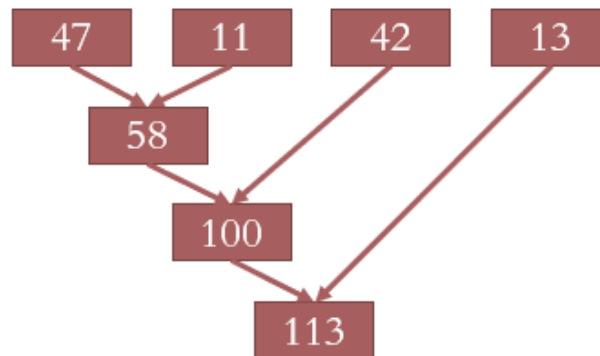
Command

```
firstUpper_UDF = func.udf(lambda s: s[0].upper() + s[1:], "STRING")  
# Apply function to our DataFrame containing the Corn Data.  
df_corn.select(firstUpper_UDF("crop").alias("Crop")).show(10)
```

Result

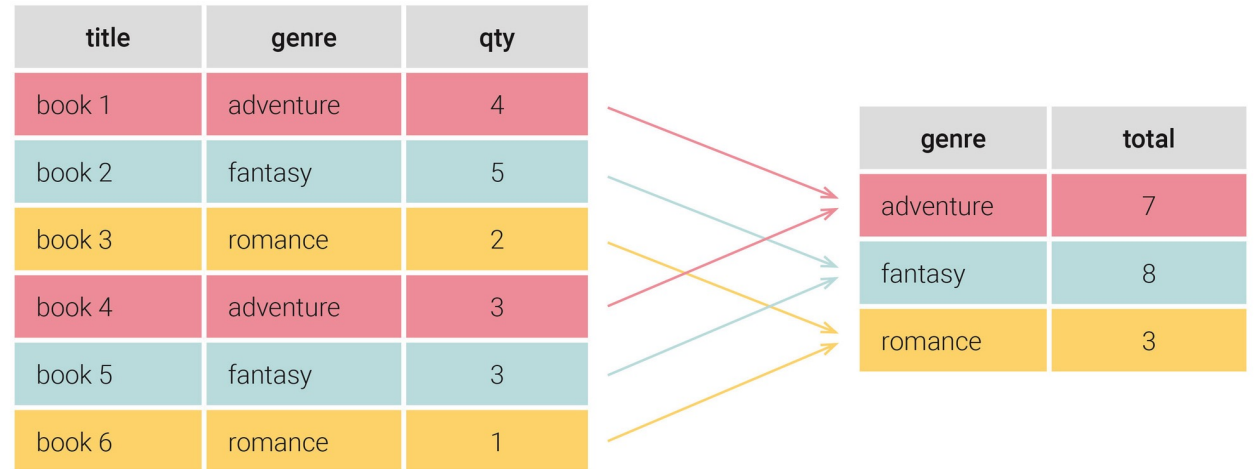
```
+-----+  
|  Crop|  
+-----+  
|Barley|  
|Barley|  
|Barley|  
|Barley|  
|Barley|  
|Barley|  
|Barley|  
|Barley|  
|  Corn|  
|  Corn|  
|  Corn|  
+-----+
```

`x = lambda a : a + 10 [47,11,42,13])`
`x(5)`



GROUPING VALUES BY ATTRIBUTE

- Rows are grouped with the same value of the group or range
- Groupby can be extended by aggregation functions
- Possible aggregate methods:
 - Sum, Min, Max, Count
 - Orderby asc, desc



GROUPING VALUES BY ATTRIBUTE

Dataset

crop	field	revenue	water_consumption	week	yield_per_sqm
barley	8	0	12	1	0
barley	8	0	10	2	0
barley	8	0	12	3	0
barley	8	35	14	4	10
barley	8	50	14	5	25
barley	8	40	18	6	15
barley	8	60	12	7	30
corn	10	0	10	1	0
corn	10	0	12	2	0
corn	10	30	20	3	25

1.

Command

```
(df_corn.select("revenue", "crop")
  .where(func.col("revenue").isNotNull())
  .groupBy("crop")
  .agg(func.round(func.mean("revenue"), 2)
    .alias("mean_rv"),
    func.max("revenue"))
  .orderBy("mean_rv", ascending=False)
  .show(n=3))
```

2.



Species Types

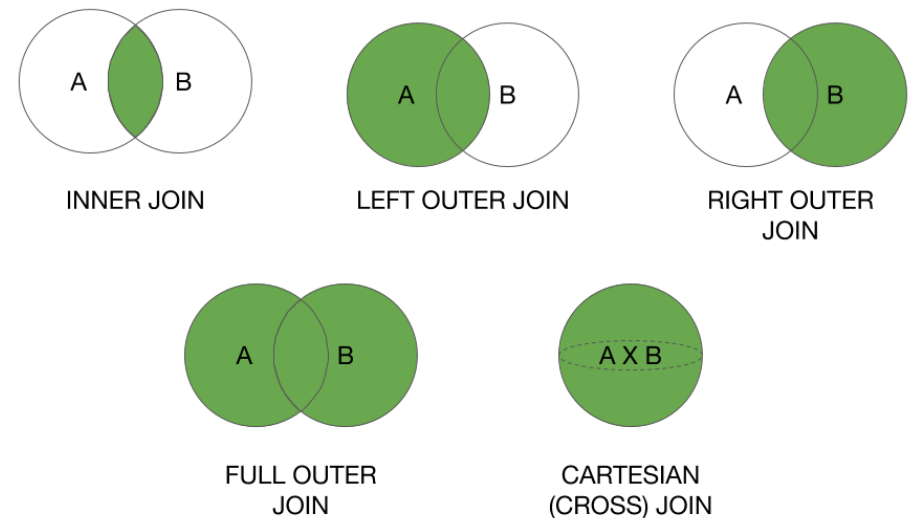
oats
corn
barley

Result

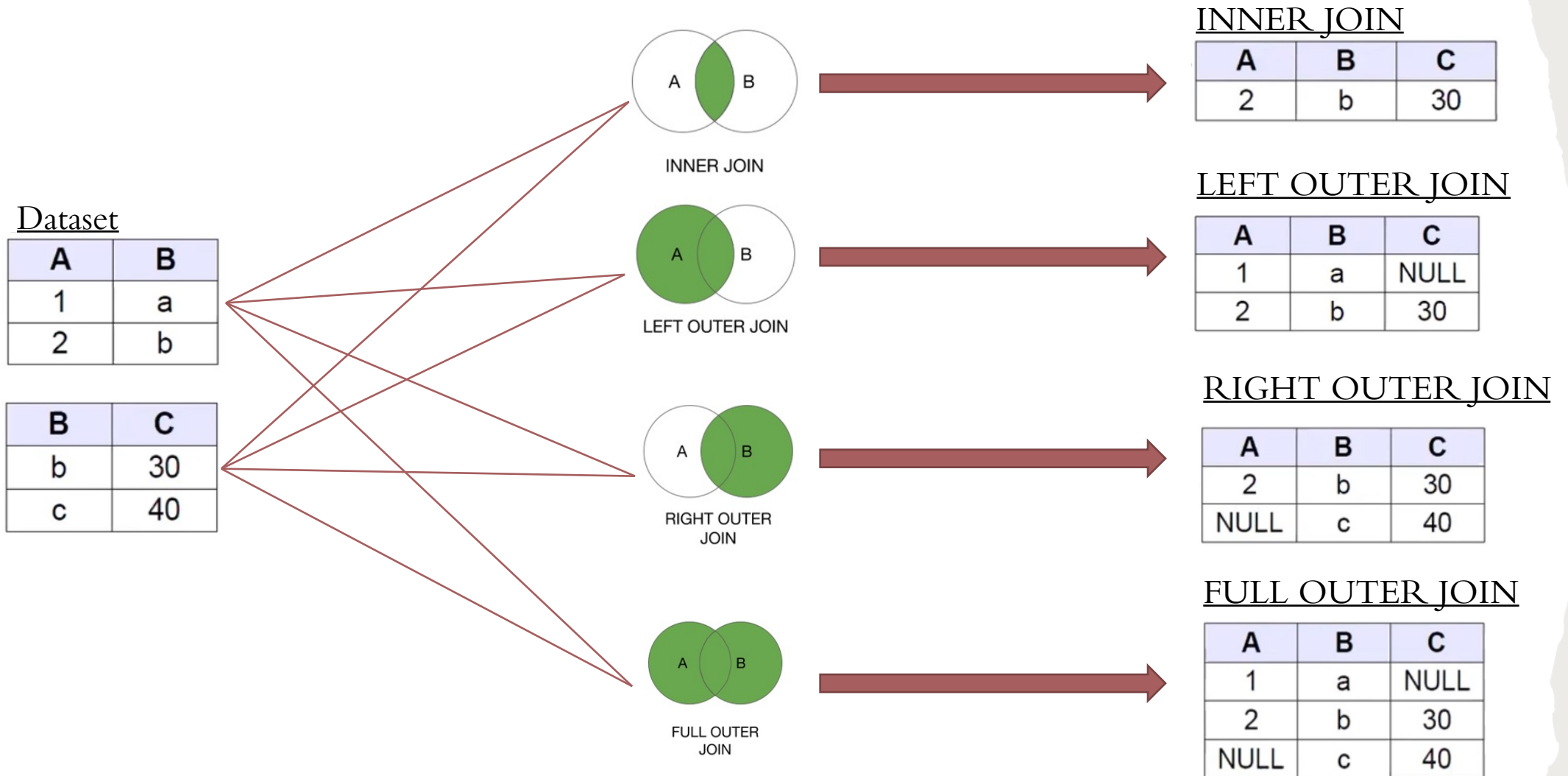
crop	mean_rv	max(revenue)
oats	95.71	150
corn	70.0	150
barley	26.43	60

JOINING DATAFRAMES

- There are several ways to connect two tables
- Differentiation between INNER and OUTER JOIN
- INNER JOIN (Natural Join)
Combination of records that meet join condition
- OUTER JOIN Association of records to which there are no correspondences of the values in the two tables



JOINING DATAFRAMES



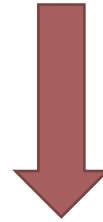
JOINING DATAFRAMES

Dataset

field_id	field_name	area_in_sqm
5	Barn Ground	10
6	Bank	10
7	Far Brossler	20
8	Middle Broom	20
9	Chalks	25
10	Big Broom	60

Command

```
gesamt_join = df_fru_veg.join(df_fields, df_fru_veg.field == df_fields.field_id,"inner")  
gesamt_join.show(30)
```



Result

crop	field	revenue	wate	crop	field	week	water_consumption	revenue	yield_per_sqm	field_id	field_name	area_in_sqm
barley	8	0		zucchini	5	0	7	4	0	5	Barn Ground	10
barley	8	0		zucchini	5	8	9	6	10	5	Barn Ground	10
barley	8	0		zucchini	5	40	5	10	25	5	Barn Ground	10
barley	8	35		zucchini	5	16	10	7	25	5	Barn Ground	10
barley	8	50		zucchini	5	40	null	11	0	5	Barn Ground	10
barley	8	40		zucchini	5	40	13	8	25	5	Barn Ground	10
barley	8	60		zucchini	5	40	12	9	25	5	Barn Ground	10
corn	10	0		zucchini	5	0	5	2	0	5	Barn Ground	10
corn	10	0		zucchini	5	0	6	3	0	5	Barn Ground	10
corn	10	30		zucchini	5	0						