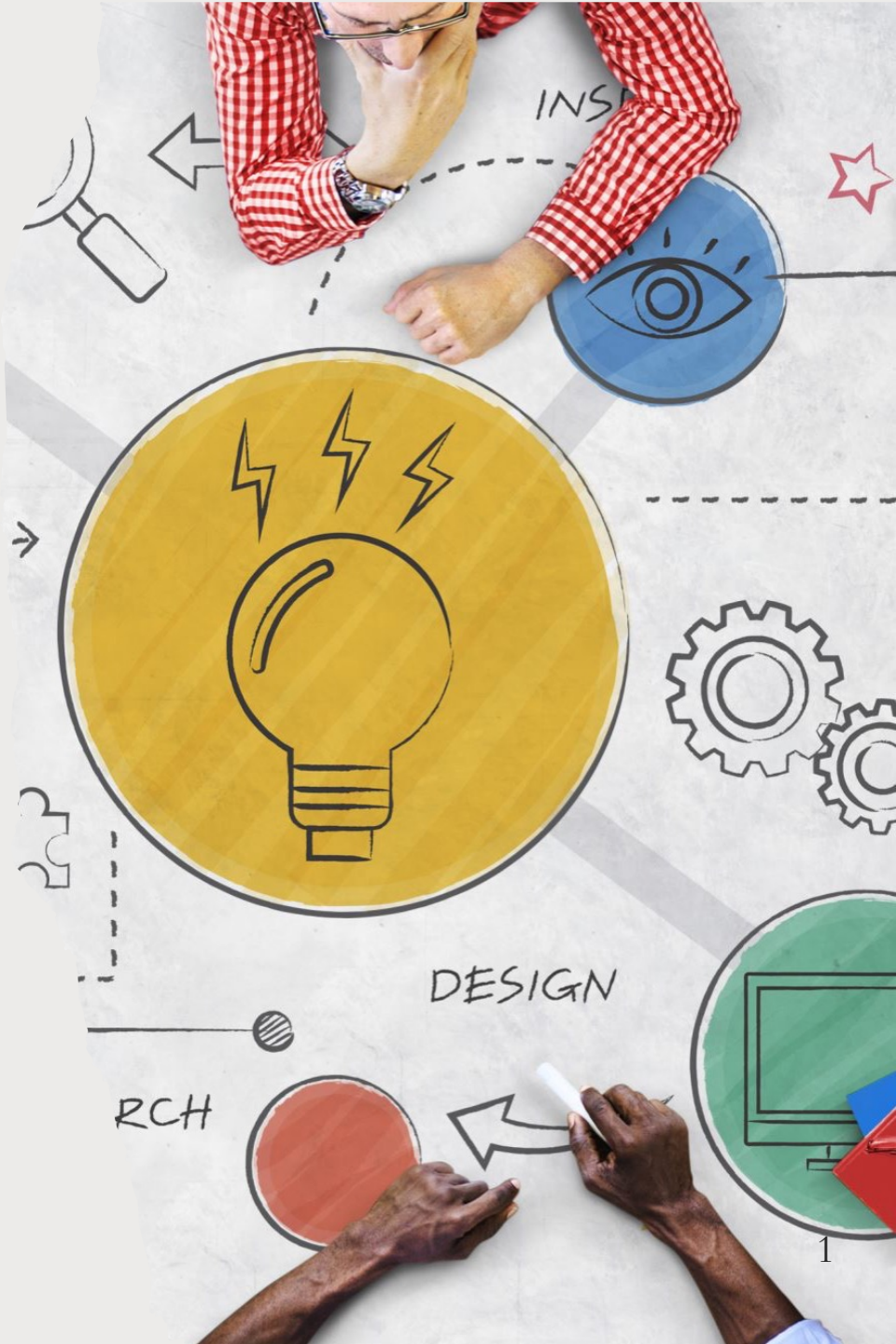
FUNCTIONS AND ADVANCED TRANSFORMATIONS



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 contraining the Iris data.
df_corn.select(firstUpper_UDF("crop").alias("Crop")).show(10)
```

Result

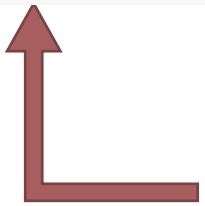
```
+------+
|  Crop|
+------+
|Barley|
|Barley|
|Barley|
|Barley|
|Barley|
|Barley|
|Barley|
|Barley|
|  Corn|
|  Corn|
|  Corn|
+------+
```

- 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.

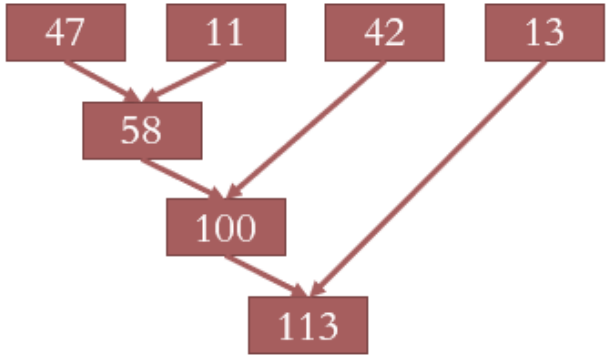
USER DEFINED FUNCTIONS

Command

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



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



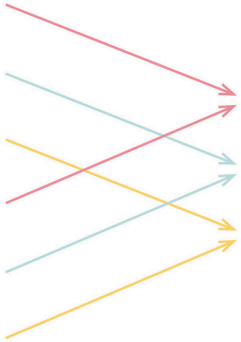
Result

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

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

title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1



genre	total
adventure	7
fantasy	8
romance	3

https://learnsql.com/blog/group-by-in-sql-explained/

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.

Species Types

oats
corn
barley

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))
```

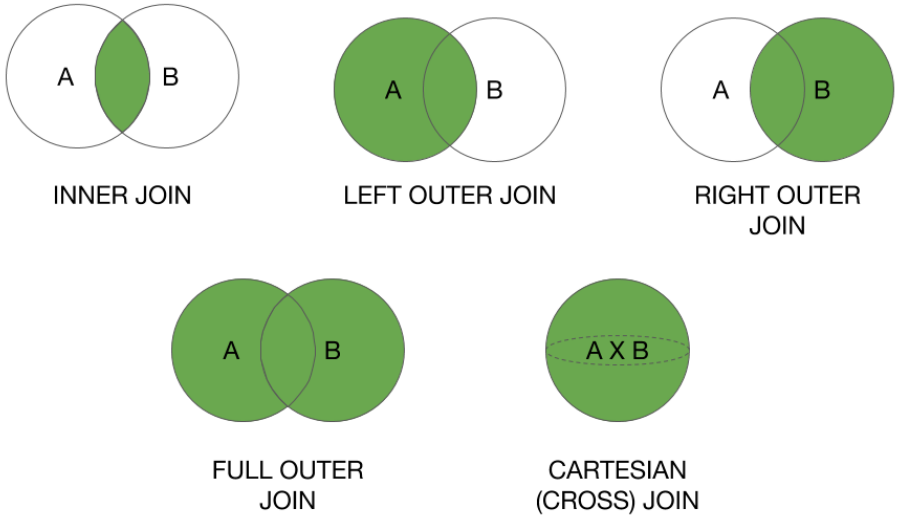
2.

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

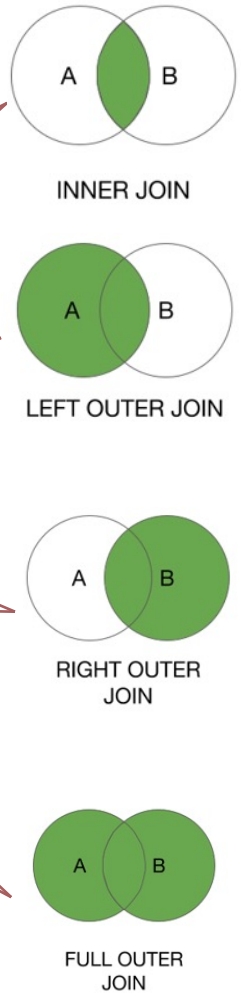


JOINING DATAFRAMES

Dataset

A	B
1	a
2	b

B	C
b	30
c	40



INNER JOIN

A	B	C
2	b	30

LEFT OUTER JOIN

A	B	C
1	a	NULL
2	b	30

RIGHT OUTER JOIN

A	B	C
2	b	30
NULL	c	40

FULL OUTER JOIN

A	B	C
1	a	NULL
2	b	30
NULL	c	40

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										