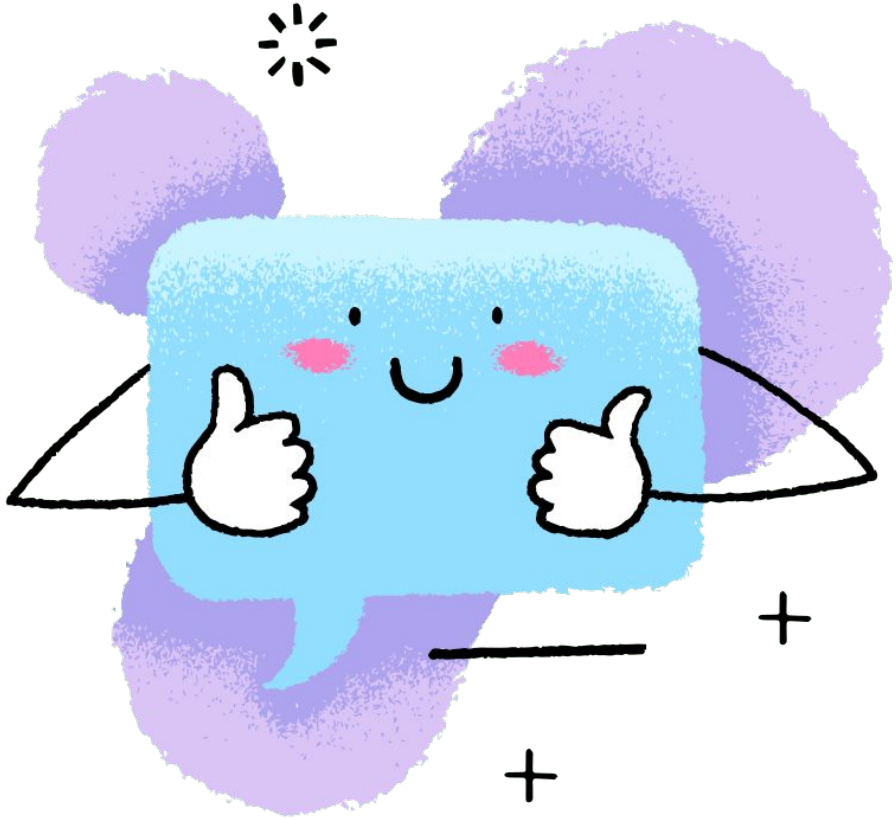
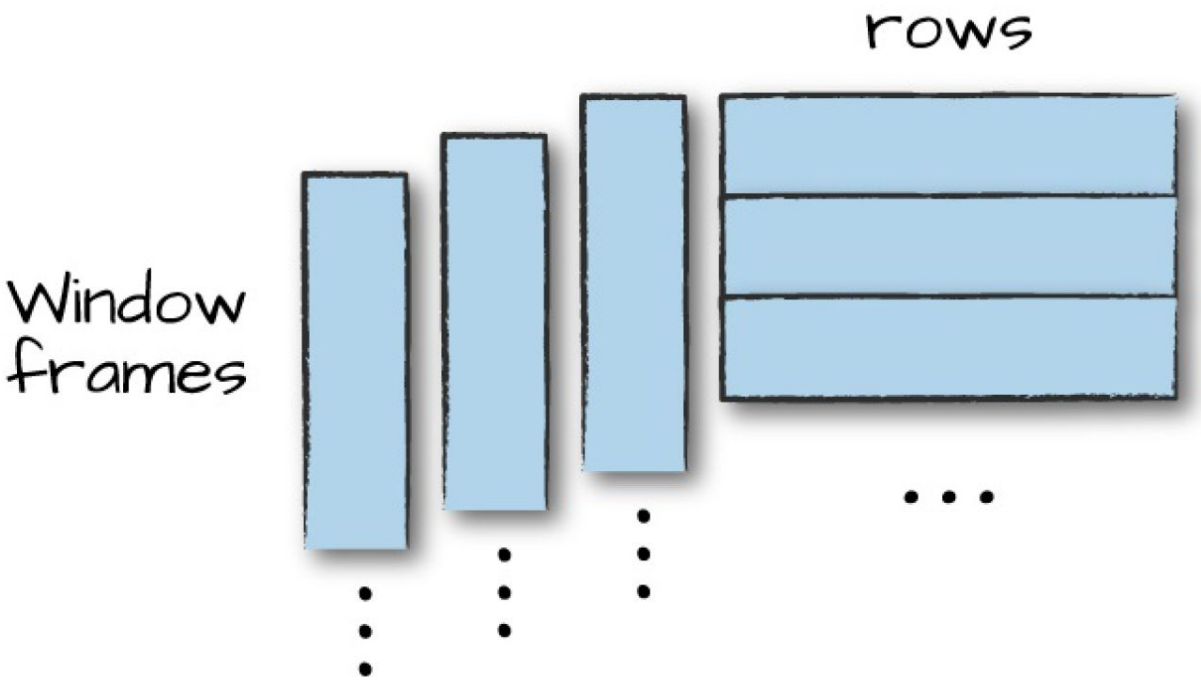
PySpark



Что будет на уроке

1. Концепция окна, что такое оконные функции
2. Как запустить оконную функцию распределенно
3. Что такое UDF, зачем они нужны

Window functions



Оконные функции в pandas

https://pandas.pydata.org/pandas-docs/stable/reference/window.html

Оконные функции в map-reduce

Как реализовать оконную функцию на чистом map-reduce?

1. Делаем на редьюсере кастомным кодом
2. Как должны быть отсортированы значения?

Window functions

WINDOW FUNCTIONS USAGE & SYNTAX	PYSPARK WINDOW FUNCTIONS DESCRIPTION
row_number(): Column	Returns a sequential number starting from 1 within a window partition
rank(): Column	Returns the rank of rows within a window partition, with gaps.
percent_rank(): Column	Returns the percentile rank of rows within a window partition.
dense_rank(): Column	Returns the rank of rows within a window partition without any gaps. Where as Rank() returns rank with gaps.
ntile(n: Int): Column	Returns the ntile id in a window partition
cume_dist(): Column	Returns the cumulative distribution of values within a window partition
lag(e: Column, offset: Int): Column lag(columnName: String, offset: Int): Column lag(columnName: String, offset: Int, defaultValue: Any): Column	returns the value that is `offset` rows before the current row, and `null` if there is less than `offset` rows before the current row.
lead(columnName: String, offset: Int): Column lead(columnName: String, offset: Int): Column lead(columnName: String, offset: Int, defaultValue: Any): Column	returns the value that is `offset` rows after the current row, and `null` if there is less than `offset` rows after the current row.

Booleans

Фильтрация по значению в колонке

```python
# in Python
from pyspark.sql.functions import col
df.where(col("InvoiceNo") != 536365)\
.select("InvoiceNo", "Description")\
.show(5, False)
```

```python
from pyspark.sql.functions import instr

priceFilter = col("UnitPrice") > 600
descripFilter = instr(df.Description, "POSTAGE") >= 1

df.where(df.StockCode.isin("DOT"))\
.where(priceFilter | descripFilter).show()
```

Регулярные выражения

```
1 from pyspark.sql.functions import regexp_replace
2 regex_string = "BLACK|WHITE|RED|GREEN|BLUE"
3 df.select(
4     regexp_replace(col("Description"), regex_string, "COLOR")
5         .alias("color_clean"),
6     col("Description")
7 ).show(2)
```

Nulls

первое не-null значение из списка столбцов

```
1 from pyspark.sql.functions import coalesce
2
3 df.select(coalesce(col("Description"), col("CustomerId")))\
4     .show()
```

удаление строк, содержащих null

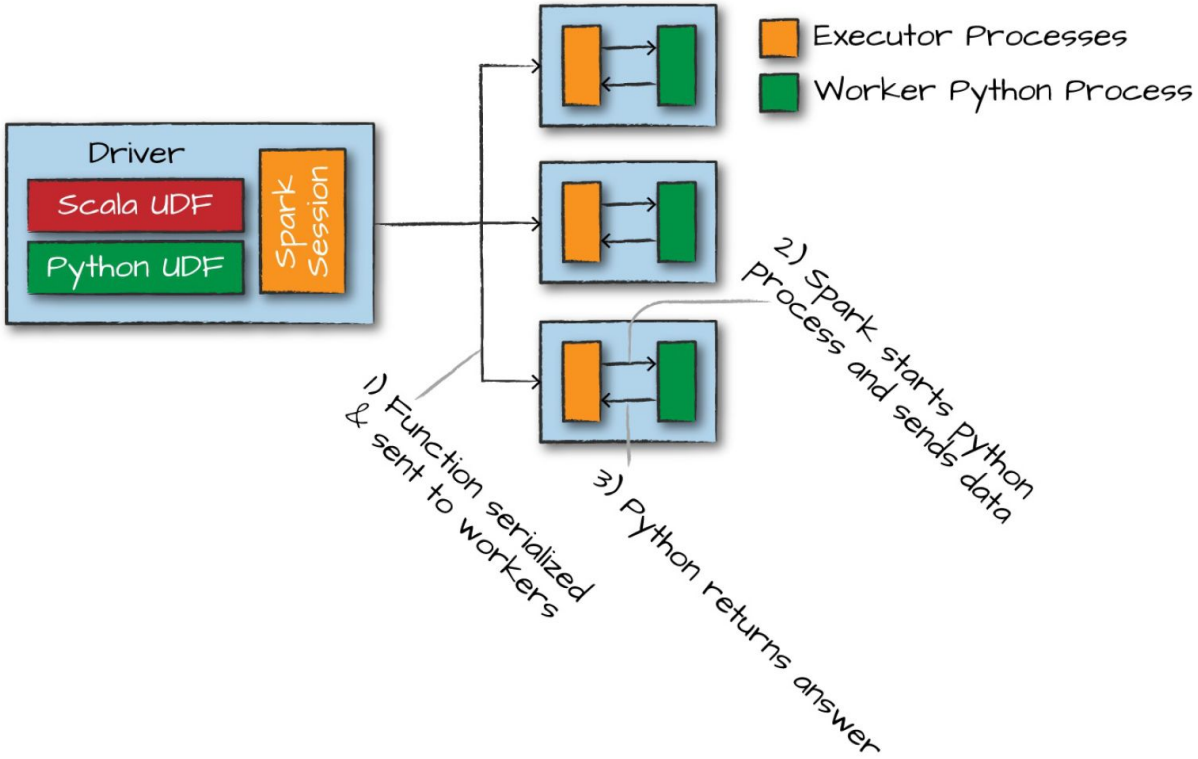
```
1 df.na.drop()
2 df.na.drop("any") # drop if ANY column is null
3
4 df.na.drop("all") # drop if ALL columns are null
5
6 df.na.drop("all", subset=["StockCode", "InvoiceNo"])
```

Nulls

заполнение пустых значений

```
1 df.na.fill("it was null value")
2
3 # specify values with dict
4 fill_cols_vals = {"StockCode": 5, "Description" : "No Value"}
5 df.na.fill(fill_cols_vals)
```

User-Defined Functions



https://docs.databricks.com/spark/latest/spark-sql/udf-python.html

Performance concerns with UDFs

- UDFs are black-box to Spark optimizations.
- UDFs block many spark optimizations like
  - WholeStageCodegen
  - Null Optimizations
  - Predicate Pushdown
  - More optimizations from Catalyst Optimizer
- String Handling within UDFs
  - UTF-8 to UTF-16 conversion. Spark maintains string in UTF-8 encoding versus Java runtime encodes in UTF-16.
  - Any String input to UDF requires UTF-8 to UTF-16 conversion.
  - Conversely, a String output requires a UTF-16 to UTF-8 conversion.

Спасибо!

Каждый день вы становитесь лучше :)