Functions

In the previous video, we've used a number of functions to manipulate our dataframe. Let's take a look at the different type of functions and their potential pitfalls.

General functions

We have used the following general functions that are quite similar to methods of Pandas dataframes:

- select(): returns a new dataframe with the selected columns
- filter(): filters rows using the given condition
- where(): is just an alias for filter()
- groupBy(): groups the DataFrame using the specified columns, so we can run aggregation on them
- sort(): returns a new DataFrame sorted by the specified column(s). By default the second parameter 'ascending' is True
- dropDuplicates(): returns a new dataframe with unique rows based on all or just a subset of columns
- withColumn(): returns a new DataFrame by adding a column or replacing the existing column that has the same name. The first parameter is the name of the new column, the second is an expression of how to compute it

Aggregate functions

Spark SQL provides built-in methods for the most common aggregations such as count(), countDistinct(), avg(), max(), min(), etc. in the pyspark.sql.functions module. These methods are not the same as the built-in methods in the Python Standard Library, where we can find min() for example as well, hence you need to be careful not to try to use them interchangeably.

In many cases, there are multiple ways to express the same aggregations. For example, if we would like to compute one type of aggregate for one or more columns of the dataframe we can just simply chain the aggregate method after a groupBy(). If we would like to use different functions on different columns agg() comes in handy. For example `agg({"salary": "avg", "age": "max"})` computes the average salary and maximum age.

User defined functions (UDF)

In Spark SQL we can define our own functions with the udf method from the pyspark.sql.functions module. The default type of the returned variable for UDFs is string. If we would like to return an other type we need to explicitly do so by using the different types from the pyspark.sql.types module.

Window functions

Window functions are a way of combining the values of ranges of rows in a dataframe. When defining the window we can choose how to sort and group (with the partitionBy method) the rows and how wide of a window we'd like to use (described by rangeBetween or rowsBetween).

For further information see the **Spark SQL, DataFrames and Datasets Guide** and the **Spark Python API Docs**.