# 01-Reading-Data

IPython has a set of predefined ‘magic functions’ that you can call with a command line style syntax. There are two kinds of magics, line-oriented and cell-oriented. Line magics are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. Cell magics are prefixed with a double %%, and they are functions that get as an argument not only the rest of the line, but also the lines below it in a separate argument.

dus %fs head /mnt/training-msft/initech/Product.csv laat de eerste regels van een bestand zien

Het bestand Product.csv staat in een Azure Blob Store container van Databricks. In 00g-Setup-DBFS is deze gemount op het Databricks File System (DBFS). Hierdoor kun je de bestanden in deze Azure Blob Strorage container benaderen alsof ze bij Databricks staan.

Je kunt eventueel laten zien dat je met Azure Storage Explorer bij deze blob container kunt door te connecten met account **databrickstraining** en key BXOG8lPEcgSjjlmsOgoPdVCpPDM/RwfN1QTrlXEX3oq0sSbNZmNPyE8By/7l9J1Z7SVa8hsKHc48qBY1tA/mgQ==

**DataFrame**

De term DataFrame komt uit Python.

# 02-Reading-Data

Mogelijke oplossing voor de bonus vraag:

csvFile = "dbfs:/mnt/training-msft/initech/Product.csv"

from pyspark.sql.types import \*

csvSchema = StructType([

StructField("product\_id", IntegerType()),

StructField("category", StringType()),

StructField("brand", StringType()),

StructField("model", StringType()),

StructField("price", DoubleType()),

StructField("processor", StringType()),

StructField("size", StringType()),

StructField("display", StringType())

])

newProductDF = (spark.read

.schema(csvSchema)

.csv(csvFile)

)

newProductDF.printSchema()

# 03-Transformations-Actions

Spark functies: https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.functions$ (wijst de link ook naar toe in Cmd 9).

spark.read: spark is de [Spark sessie](https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.SparkSession). read is een Value Member die een [DataFrameReader](https://spark.apache.org/docs/latest/api/scala/org/apache/spark/sql/DataFrameReader.html) teruggeeft. Dat is weer een interface om data van externe systemen te lezen. Je krijgt dan een *untyped view*, een DataFrame. Daar hoort weer een strongly typed [DataSet](https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.Dataset) bij. En daar horen weer Actions bij, zoals **show**.

Het Azure Databricks Notebook wat je gebruikt, heeft ook weer functies, waaronder **display**. Zie <https://docs.azuredatabricks.net/user-guide/visualizations/index.html>.

Apache Spark pyspark.sql documentatie: <https://spark.apache.org/docs/latest/api/python/pyspark.sql.html>. Hier vind je bijv. ook de functies die in deze module zitten en waarvan de **col** functie in Cmd 31 wordt gebruikt.

De print functie in Cmd 21 is een Python functie. Op de plek van de {} wordt het argument wat je via .format opgeeft getoond. {:,} betekend weer duizendtallen scheiding (wat hier niet nodig is).

Cmd 34/35: Column: <https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.Column>

Cmd 46/47: verschil tussen limit, take en collect?

* take(n) geeft een **lijst van n Row** objecten in een DataFrame terug
* collect() geeft een **lijst van alle Row** objecten in een DataFrame terug
* limit(n) beperkt een **DataFrame** tot n records (maar resultaat blijft een DataFrame)

Cmd 48/49: “the driver” = de master node, die alle data ontvangt

.