CHEATSHEET: PANDAS VS PYSPARK

Vanessa Afolabi

Import Libraries and Set System Options:

PANDAS	PYSPARK

import pandas as pd
pd.options.display.max_colwidth = 1000
	from pyspark.sql.types import *
	from pyspark.sql.functions import *
	from pyspark.sql import SQLContext*

Define and create a dataset:

PANDAS	PYSPARK

data = {'col1' : [, ,], 'col2' : [, ,]}
df = pd.DataFrame(data, columns = ['col1', 'col2'])
	StructField('Col1', IntegerType())
	StructField('Col2', StringType())
	schema = StructType([list of StructFields])
	df = SQLContext(sc).createDataFrame(sc.emptyRDD(), schema)

Read and Write to CSV:

PANDAS	PYSPARK
df.read_csv()	SQLContext(sc).read_csv()
df.to_csv()	df.toPandas.to_csv()

Indexing and Splitting:

PANDAS	PYSPARK
df.loc[ ] df.iloc[ ]	df.randomSplit(weights=[ ], seed=n)

Inspect Data:

PANDAS	PYSPARK

df.head()	df.show()
	df.head(n)

df.columns	df.printSchema() df.columns
df.shape	df.count()

Handling Duplicate Data:

PANDAS	PYSPARK
df.unique() df.duplicated	df.distinct().count()
df.drop_duplicates()	df.dropDuplicates()

Rename Columns:

PANDAS	PYSPARK
df.rename(columns={"old_col":"new_col"})	df.withColumnRenamed("old_col","new_col")

Handling Missing Data:

PANDAS	PYSPARK
df.dropna()	df.na.drop()
df.fillna()	df.na.fill()
df.replace	df.na.replace()
df['col'].isna() df['col'].isnull()	df.col.isNull()
df['col'].notna() df['col'].notnull()	df.col.isNotNull()

Common Column Functions:

PANDAS	PYSPARK
df["col"] = df["col"].str.lower()	df = df.withColumn('col',lower(df.col))

df["col"] = df["col"].str.replace()	df = df.select('*',regexp_replace().alias())
	df = df.select('*',regexp_extract().alias())

df["col"] = df["col"].str.split()	df = df.withColumn('col',split('col'))
df["col"] = df["col"].str.join()	df = df.withColumn('col', UDF_JOIN(df.col, lit(' ')))
df["col"] = df["col"].str.strip()	df = df.withColumn('col', trim(df.col))

Apply User Defined Functions:

PANDAS	PYSPARK

df['col'] = df['col'].map(UDF)
df.apply(f)
df.applyMap(f)
	df = df.withColumn('col', UDF(df.col))
	df = df.withColumn('col', when(cond, UDF(df.col)).otherwise())

Join two dataset columns:

PANDAS	PYSPARK

df['new_col'] = df['col1'] + df['col2']	df = df.withColumn('new_col',concat_ws(' ',df.col1,df.col2))
	df.select('*',concat(df.col1,df.col2).alias('new_col'))

Convert dataset column to a list:

PANDAS	PYSPARK
list(df['col')	df.select("col").rdd.flatMap(lambda x:x).collect()

Filter Dataset:

PANDAS	PYSPARK

df = df[df['col'] != " "]	df = df[df['col'] == val]
	df = df.filter(df['col'] == val)

Select Columns:

PANDAS	PYSPARK
df = df[['col1','col2','col3']]	df = df.select('col1','col2','col3')

Drop Columns:

PANDAS	PYSPARK

df.drop(['B','C'], axis=1)
df.drop(columns = ['B','C'])
	df.drop('col1','col2')

Grouping Data:

PANDAS	PYSPARK
df.groupby(by=['col1','col2']).count()	df.groupBy('col').count().show()

Combining Data:

PANDAS	PYSPARK

pd.concat([df1,df2])
df1.append(df2)
	df1.union(df2)

df1.join(df2)	df1.join(df2)

Cartesian Product:

PANDAS	PYSPARK
df1['key'] = 1 df2['key'] = 1 df1.merge(df2, how='outer', on='key')	df1.crossJoin(df2)

Sorting Data:

PANDAS	PYSPARK

df.sort_values()
df.sort_index()
	df.sort()
	df.orderBy()