Pandas and PySpark Compression Statement:

**By- Tinendra Kumar (Data Engineer @ CHUBB)**



**Table of Contents:**   
**1.**[**Create/Read/Write files as Dataframes**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#a809) **2.**[**Inspect Data**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#506c) **3.**[**Filter/Queries**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#8bd4) **4.**[**Drop or Fill Nulls and Duplicate Values**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#8e32) **5.**[**Pivot/Sort/Join Dataframe**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#c3bd) **6.**[**User Defined Functions (UDF)**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#d1e5) **7.**[**Group By**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#c77c) **8.**[**Window Functions**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#017c) **9.**[**Add/Update/Remove/Rename/Typecast Columns**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#2117) **10.**[**Miscellaneous**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#c6c8) **🡪 FOR MORE STAYS TUNED ….**

1. [**Create/Read/Write files as Dataframes**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#a809)**:**

**# Create Dataframe from list of tuples**

**data = [(1,"a"), (2,"b")]**

**columns = ["id", "key"]**

**df = pd.DataFrame(data=data, columns = columns) # In pandas**

**sdf = spark.createDataFrame(data, columns) # In spark**

**# Create Dataframe from list of dictionaries**

**data = [{"id":1,"key":"a"}, {"id":2,"key":"b"}]**

**df = pd.DataFrame(data=data) # In pandas**

**sdf = spark.createDataFrame(data) # In spark**

**# Convert spark dataframe into pandas dataframe**

**df = sdf.toPandas()**

**# Convert pandas dataframe into spark datafrrame**

**sdf = spark.createDataFrame(df)**

**# Reading a csv file**

**df = pd.read\_csv(file\_path # In pandas**

**sdf = spark.read.csv(file\_path) # In spark**

**# Writing a csv file**

**df.to\_csv(file\_path) # In pandas**

**sdf.write.csv(file\_path) # In spark**

1. [**Inspect Data**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#506c)**:**

**# Display the content of df (n rows)**

**sdf.limit(n).show() # In spark**

**df.head(n) # In pandas**

**# Return df column names and data types**

**df.dtypes # In spark and pandas**

**# Return the columns of df**

**df.columns # In spark and pandas**

**# Count the number of rows in df**

**len(df) # In pandas**

**sdf.count() # In spark**

**# Compute summary statistics**

**df.describe() # In pandas**

**df.describe().show() # In spark**

1. [**Filter/Queries**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#8bd4)**:**

**# Filter on one colum**

**df = df[df.col > value] # In pandas**

**sdf = sdf.filter(sdf.col > value) # In spark**

**sdf = sdf[sdf.col.isin(["val1","val2"])] # In spark and pandas**

**# Filter on multiple columns**

**df = df[(df.col1==val1) & (df.col2==val2)] # In pandas**

**sdf = sdf.filter((sdf.col1==val1) & (sdf.col2==val2)) # In spark**

**NOTE : use '&' for 'and', '|' for 'or', '~' for 'not' when building DataFrame boolean expressions.**

**# select query**

**df = df[['col1', 'col2']] # In pandas**

**sdf = sdf.select(col1, col2) # In spark**

**from pyspark.sql import functions as F**

**sdf =sdf.select("col", F.when(sdf.col > val, val\_if\_true)**

**.otherwise(val\_if\_false)) # In spark**

1. [**Drop or Fill Nulls and Duplicate Values**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#8e32)**:**

**# Drop Duplicate Values**

**df = df.drop\_duplicates(subset = ["col\_1"]) # In pandas**

**sdf = sdf.dropDuplicates(subset = ["col\_1"]) # In spark**

**# Remove Nulls**

**df = df.dropna() # In spark and pandas**

**# Fill nan values**

**df = df.fillna(value) # In spark and pandas**

**df = df.fillna({col\_1:value\_1,col\_2:value\_2} # In pandas**

**sdf = sdf.fillna(value, subset = ["col\_1"]) # In spark**

1. [**Pivot/Sort/Join Dataframe**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#c3bd)**:**

**# Sort Dataframe**

**df = df.sort\_values(by=['col1','col2'],ascending=False) # In pandas**

**sdf = sdf.orderBy(["col\_1","col\_2"],ascending=[0,1]) # In spark**

**sdf = df.sort("col", ascending=False)**

**# Pivot Dataframe**

**df = df.pivot\_table(index='col1', columns='col2',**

**values='col3', aggfunc=sum) # In pandas**

**sdf = sdf.groupBy("col1").pivot("col2").sum("col2") # In spark**

**# Join Dataframes**

**df = df1.join(df2, on = 'column\_name',**

how = 'join\_type') # In spark and pandas

1. [**User Defined Functions (UDF)**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#d1e5)**:**

**# UDF with single column**

**df['squared\_col'] = df['col'].apply(lambda x:x\*x) # In pandas**

**from pyspark.sql import functions as F**

**squared = F.udf(lambda x: x\*x, IntegerType())**

**sdf = sdf.withColumn("squared\_col", squared('col')) # In spark**

**# UDF with multiple column**

**df['col1\*col2'] = df.apply(lambda x : x[col1]\*x[col2],**

**axis=1) # In pandas**

**def multiply(x):**

**x['col1\*col2'] = x[col1]\*x[col2]**

**return x**

**df = df.apply(multiply, axis=1) # In pandas**

**from pyspark.sql import functions as F**

**squared = F.udf(lambda x,y: x\*y, IntegerType())**

**sdf = sdf.withColumn("col1\*col2",**

**squared('col1', 'col2')) # In spark**

1. [**Group By**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#c77c)**:**

**## Pandas**

**# Aggregation with single column**

**squared = lambda x : x\*x**

**df = df.groupby("col").agg({"col1": squared, "col2":'count'})**

**# Aggregation with multiple column column**

**def avg\_sum(x):**

**x["avg\_sum"] = np.mean(x.col1 + x.col2)**

**return x**

**df = df.groupby("col").apply(avg\_sum)**

**## Spark**

**# Aggregation with single column (inbuilt functions)**

**sdf = sdf.groupBy("col")**

**.agg(F.count("col1").alias("col1\_count"),**

**F.count("col2").alias("col2\_count"))**

**# Aggregation with multiple columns (user defined functions)**

**import numpy as np**

**@F.pandas\_udf(DoubleType(),**

**functionType=F.PandasUDFType.GROUPED\_AGG)**

**def f(x, y):**

**return np.mean(x+y)**

**sdf = sdf.groupBy("col").agg(f("col1", "col2").alias("avg\_sum"))**

1. [**Window Functions**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#017c)**:**

**# Window with in built spark functions**

**from pyspark.sql.window import Window**

**win = Window().partitionBy('col1')**

**.orderBy(F.col('col2').desc())**

**sdf = sdf1.withColumn('row\_num', F.row\_number().over(win))**

**# Window with in user defined functions in spark**

**@F.pandas\_udf(ArrayType(IntegerType()),**

**F.PandasUDFType.GROUPED\_AGG)**

**def get\_running\_mean(df:pd.Series):**

**return np.mean(df)**

**w = Window.partitionBy('col1').orderBy('col2')**

**sdf = sdf.withColumn('running\_mean',**

**get\_running\_mean('col').over(w))**

**# Window with in built pandas functions**

**df['col\_rolling\_sum'] = df['col'].rolling(window=3,**

**min\_periods=1).sum()**

**# Window with in user defined functions in pandas**

**df['col2'] = df['col1'].rolling(window=3, min\_periods=1)**

**.apply(lambda x : np.mean(x))**

1. [**Add/Update/Remove/Rename/Typecast Columns**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#2117)**:**

**# Add Columns**

**df["column\_name"] = "Any literal" or dataframe column # In pandas**

**from pyspark.sql import functions as F # In spark**

**sdf=sdf.withColumn('column\_name',F.lit("any literal"))\**

**.withColumn("column\_name\_2', sdf.column\_name)**

**# Remove Columns**

**df = df.drop(columns=["column\_1", "column\_2") # In pandas**

**sdf = sdf.drop("column\_1", "column\_2") # In spark**

**# Rename one Column**

**df.rename(columns={'old\_col':'new\_col'},inplace=True) # In pandas**

**sdf = sdf.withColumnRenamed('old\_col', 'new\_col') # In spark**

**# Rename all Columns**

**df.columns = ["col\_1", "col\_2", ... "col\_n"] # In pandas**

**sdf = sdf.toDF("col\_1", "col\_2", ... "col\_n") # In spark**

**# Typecast one column**

**df.astype({'col1': column\_type}) # In pandas**

**from pyspark.sql.types import \* # In spark**

**sdf = sdf.withColumn("dd", sdf.dd.cast(column\_type))**

**# Typecast multiple columns**

**df.astype(column\_type) # In pandas**

**sdf = sdf.select([sdf.column.cast(column\_type) for**

**column in sdf.columns]) # In spark**

**10.**[**Miscellaneous**](https://medium.com/@rahulnkumar/pandas-to-pyspark-33269e8607c3#c6c8)**:**

**# Concat two dataframes columnwise in spark**

**from pyspark.sql import functions as F**

**sdf1 = sdf1.withColumn("id", F.monotonically\_increasing\_id())**

**sdf2 = sdf2.withColumn("id", F.monotonically\_increasing\_id())**

**sdf3 = sdf2.join(sdf1, "id", "outer").drop("id")**

**# Concat two dataframes rowwise in spark**

**sdf3 = sdf1.union(sdf2)**

**# Concat two dataframes columnwise in pandas**

**df3 = pd.concat([df1, df2], axis=1)**

**# Concat two dataframes rowwise in pandas**

**df3 = pd.concat([df1, df2], axis=0)**

**# column to list in spark (two ways)**

**col = sdf.select("col").rdd.map(lambda x: x[0]).collect()**

**col = sdf.select("col").rdd.flatMap(lambda x: x).collect()**

**# column to list in pandas**

**col = df.col.tolist()**