From pandas To PySpark

Read Files

pandas

```python
import pandas as pd
df = pd.read_parquet("file.parquet")
df = pd.read_csv("file.csv")
df = pd.read_json("file.json")
```

PySpark

```python
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Read Parquet").getOrCreate()
df = spark.read.parquet("file.parquet")
df = spark.read.json('file.json')
df = spark.read.csv('file.csv')
```

Description

pandas

```python
# Nombre de lignes
num_rows = len(df)
# Résumé stats
df.describe()
# Info
df.info()
```

PySpark

```python
# Nombre de lignes
num_rows = df.count()
# Résumé stats
df.describe().show()
# Info
df.printSchema()
```

Sélectionner des colonnes

pandas

```
selected_columns = df[['column1', 'column2']]
```

PySpark

```
selected_columns = df.select('column1', 'column2')
```

Ajouter une colonne

pandas

```
df['new_column'] = df['old_column'] * 2
```

PySpark

```
from pyspark.sql.functions import col
df = df.withColumn('new_column', col('old_column') * 2)
```

Renommer une colonne

pandas

```
df = df.rename(columns={'old_name': 'new_name'})
```

PySpark

```
df = df.withColumnRenamed('old_name', 'new_name')
```

Ordonner les colonnes

pandas

```
sorted_df = df.sort_values('column_name', ascending=False)
```

PySpark

```
from pyspark.sql.functions import desc
sorted_df = df.orderBy(desc('column_name'))
```

Filtrer

pandas

```
filtered_df = df[df['column_name'] > 5]
```

PySpark

```
from pyspark.sql.functions import col
filtered_df = df.filter(col('column_name') > 5)
```

Join DataFrames

pandas

```
joined_df = pd.merge(df1, df2, on='id', how='inner')
```

PySpark

```
joined_df = df1.join(df2, on='id', how='inner')
```

Concaténer

pandas

```
#Verticalement
new_df = pd.concat([df1, df2], axis=0)

#Horizontalement
new_df = pd.concat([df1, df2], axis=1)
```

PySpark

```
#Verticalement
new_df = df1.union(df2)

#Horizontalement
new_df = df1.join(df2)
```

Valeurs manquantes

pandas

```python
# Enlève toutes les valeurs manquantes
df = df.dropna()

#Remplace les valeurs manquantes par la médiane
filled_df = df.fillna(df.median())
```

```python
from pyspark.sql.functions import col, median
# Enlève toutes les valeurs manquantes
df = df.dropna()

#Remplace les valeurs manquantes par la médiane
filled_df = df.fillna(median(df.col1), subset=['col1'])
```

PySpark

Enlever doublons

pandas

```
df.drop_duplicates(subset=['col1', 'col2'], inplace=True)
```

PySpark

```
df = df.dropDuplicates(['col1', 'col2'])
```

Caster

pandas

```
df['col'] = df['col'].astype('category')
```

PySpark

```
from pyspark.sql.functions import col
df = df.withColumn('col', col('col').cast('string'))
```

Groupby & agg

pandas

```
df.groupby('col').agg({'col2': 'mean'})
```

PySpark

```
from pyspark.sql.functions import mean
grouped_df = df.groupBy('col').agg(mean('col2'))
```

Pivot Table

pandas

```
pivot_table = pd.pivot_table(df, values='value_column',
                                 index='index_column',
                                 columns='column_to_pivot')
```

PySpark

```
from pyspark.sql.functions import first
pivot_table = df.groupBy('index_column')\
                .pivot('column_to_pivot')\
                .agg(first('value_column'))
```

Remplacer des valeurs

pandas

```
df['col'].replace(['old_value1', 'old_value2'], 'new_value', inplace=True)
```

PySpark

```
from pyspark.sql.functions import when

df = df.withColumn('col', when(col('col')\
    .isin(['old_value1', 'old_value2']), 'new_value')\
    .otherwise(col('col')))
```

Filtrer avec un regex

pandas

```
selected_rows = df[df['col'].str.contains('pattern')]
```

PySpark

```
from pyspark.sql.functions import regexp_extract
selected_rows = df.filter(regexp_extract(col('col'), 'pattern', 0) != '')
```

Appliquer une fonction

pandas

```
df['new_column'] = df['old_column'].apply(lambda x: x * 2)
```

```
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

multiply_udf = udf(lambda x: x * 2, IntegerType())
df = df.withColumn('new_column', multiply_udf('old_column'))
```

PySpark

Windows function

pandas

```
df['rolling_sum'] = df.groupby('group')['value']\
                      .rolling(window=2, min_periods=1).sum()\
                      .reset_index(drop=True)
```

PySpark

```
from pyspark.sql.functions import col, sum, lag
from pyspark.sql.window import Window

w = Window.partitionBy('group').orderBy('value')
df = df.withColumn('rolling_sum', sum(col('value')).over(w.rowsBetween(-1, 0)))

df.show()
```

Output :

	group	value	rolling_sum
0	A	1	1.0
1	A	2	3.0
2	B	3	3.0
3	B	4	7.0
4	B	5	9.0

linkedin.com/in/maximetut/

Export

pandas

```
#Export csv
df.to_csv('my_dataframe.csv', index=False)
#Export parquet
df.to_parquet('my_dataframe.parquet', index=False)
#Export JSON
df.to_json('my_dataframe.json', orient='records')
```

PySpark

```
#Export csv
df.write.csv('my_dataframe.csv', header=True, mode='overwrite')
#Export parquet
df.write.parquet('my_dataframe.parquet', mode='overwrite')
#Export JSON
df.write.json('my_dataframe.json', mode='overwrite')
```

10 librairies Python pour l’extraction de données via des APIs

linkedin.com/in/maximetut/