**Class 3 PySpark & SparkSQL codings**

ở buổi trước chúng ta đã học về RDD, và làm quen với SparkSQL.

Hôm nay chúng ta sẽ đi vào chi tiết về coding trong Pyspark , và SparkSQL.

Remind 1 tí : Pyspark = Python + Spark

SparkSQL = SQL + Spark

* Pyspark là dùng Python trong Spark
* SparkSQL là dùng SQL trong Spark
* Pyspark

Đầu tiên chính là phần load data, tạo DataFrame

Tạm gọi 2 biến

Dp : Dataframe Python

Ds : Dataframe Spark

# Cách Dùng pyspark trên colab

Đầu tiên là cài đặt pyspark bằng câu lệnh !pip install pyspark

Thứ 2 là mount cái google drive của mình vào colab

from google.colab import drive

drive.mount('/content/drive/')

from pyspark.context import SparkContext

from pyspark.sql.session import SparkSession

from pyspark.sql.functions import \*

import pandas as pd

**Pandas DataFrame :**

Dp = pd.read\_csv(filepath)

**Spark DataFrame :**

Ds = spark.read.csv(filepath)

Ds = spark.read.json(filepath)

Ds = spark.read.parquet(filepath)

Ngoài ra , chúng ta có thể dùng câu lệnh cho các loại data

Ds = Spark.read.format(“csv”).option(“header”,”true”).load(filepath)

Vì cơ bản Pyspark cũng là Python , nên hầu như mọi câu lệnh của Pyspark Dataframe đều gần giống với Pandas Dataframe

**In top 10 các giá trị đầu**

Dp.head(10)

Ds.show(10)

**In danh sách các cột**

Dp.columns

Ds.columns

**In thông tin bảng**

Dp.info()

Ds.printSchema()

**Slicing Data**

Dp[[‘column1’,’column2’]]

Ds.select(‘column1’,’column2’)

**Fill Null**

my\_list = [['male', 1, None], ['female', 2, 3],['male', 3, 4]]

dp = pd.DataFrame(my\_list,columns=['A', 'B', 'C'])

ds = spark.createDataFrame(my\_list, ['A', 'B', 'C'])

Dp.fillna(0)

Ds.fillna(0)

**Replace Values**

Dp[‘A’].replace(['male', 'female'],[1, 0], inplace=True)

ds.na.replace(['male','female'],['1','0']).show()

**Rename Columns**

mapping = {'Newspaper':'C','Sales':'D'}

dp.rename(columns=mapping).head(4)

new\_names = [mapping.get(col,col) for col in ds.columns]

ds.toDF(\*new\_names).show(4)

#

ds.withColumnRenamed('Newspaper','Paper').show(4)

**Drop Columns**

drop\_name = ['Newspaper','Sales']

dp.drop(drop\_name,axis=1).head(4)

ds.drop(\*(drop\_name)).show(4)

**Filter**

dp = pd.read\_csv('Advertising.csv')

ds = spark.read.csv(path='Advertising.csv', header=True)

dp[dp.Newspaper<20].head(4)

ds[ds.Newspaper<20].show(4)

Multiple conditions

dp[(dp.Newspaper<20)&(dp.TV>100)].head(4)

ds[(ds.Newspaper<20)&(ds.TV>100)].show(4)

**With New Column**

Như cái tên , dùng để tạo ra 1 column mới

dp['tv\_norm'] = dp.TV/sum(dp.TV)

dp.head(4)

ds.withColumn('tv\_norm', ds.TV/ds.groupBy().agg(F.sum("TV")).show(4)

dp['cond'] = dp.apply(lambda c: 1 if ((c.TV>100)&(c.Radio 10 else 3,axis=1)

ds.withColumn('cond',F.when((ds.TV>100&(ds.Radio<40),1).when(ds.Sales>10, 2) .otherwise(3)).show(4)

dp['log\_tv'] = np.log(dp.TV)

dp.head(4)

import pyspark.sql.functions as F

ds.withColumn('log\_tv',F.log(ds.TV)).show(4)

dp['tv+10'] = dp.TV.apply(lambda x: x+10)

dp.head(4)

ds.withColumn('tv+10', ds.TV+10).show(4)

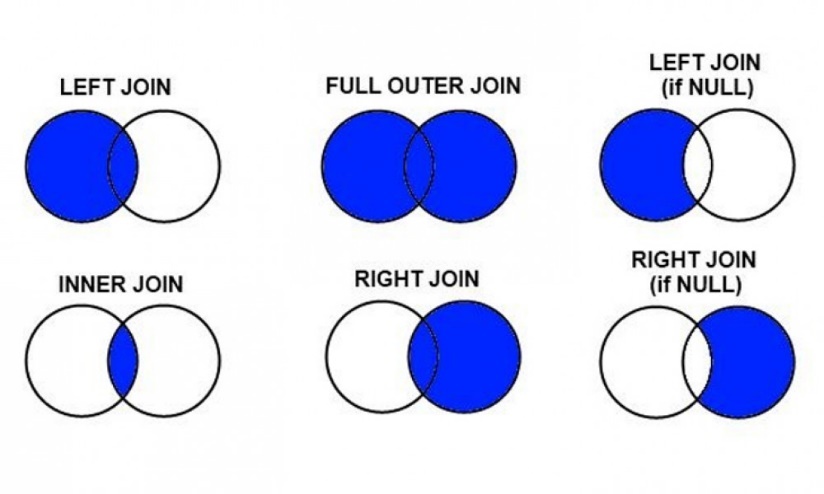
**Join**

leftp = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'], 'B': ['B0', 'B1', 'B2', 'B3'], 'C': ['C0', 'C1', 'C2', 'C3'], 'D': ['D0', 'D1', 'D2', 'D3']}, index=[0, 1, 2, 3])

rightp = pd.DataFrame({'A': ['A0', 'A1', 'A6', 'A7'], 'F': ['B4', 'B5', 'B6', 'B7'], 'G': ['C4', 'C5', 'C6', 'C7'], 'H': ['D4', 'D5', 'D6', 'D7']}, index=[4, 5, 6, 7])

lefts = spark.createDataFrame(leftp)

rights = spark.createDataFrame(rightp)



1. Left Join

leftp.merge(rightp,on='A',how='left')

lefts.join(rights,on='A',how='left') .orderBy('A',ascending=True).show()

1. Right Join

leftp.merge(rightp,on='A',how='right')

lefts.join(rights,on='A',how='right') .orderBy('A',ascending=True).show()

1. Inner Join

leftp.merge(rightp,on='A',how='inner')

lefts.join(rights,on='A',how='inner') .orderBy('A',ascending=True).show()

1. Outer Join

leftp.merge(rightp,on='A',how='outer')

lefts.join(rights,on='A',how='full') .orderBy('A',ascending=True).show()

**Concat**

my\_list = [('a', 2, 3), ('b', 5, 6), ('c', 8, 9), ('a', 2, 3), ('b', 5, 6), ('c', 8, 9)]

col\_name = ['col1', 'col2', 'col3']

dp = pd.DataFrame(my\_list,columns=col\_name)

ds = spark.createDataFrame(my\_list,schema=col\_name)

dp['concat'] = dp.apply(lambda x:'%s%s'%(x['col1'],x['col2']),axis=1)

ds.withColumn('concat',F.concat('col1','col2')).show()

**Group by**

dp.groupby(['col1']).agg({'col2':'min','col3':'mean'})

ds.groupBy(['col1']).agg({'col2': 'min', 'col3': 'avg'}).show()

**Order by**

dp.sort\_values(‘col1’,ascending = True)

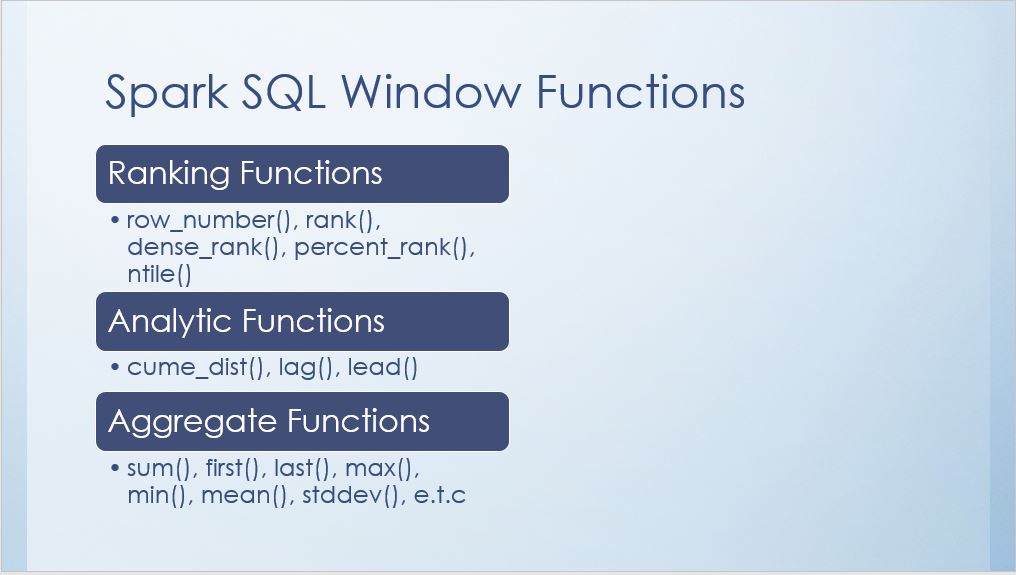
ds.orderBy(‘col1’,ascending = True)

**Pivot**

pd.pivot\_table(dp, values='col3', index='col1', columns='col2', aggfunc=np. ˓→sum)

ds.groupBy(['col1']).pivot('col2').sum('col3').show()

**Window**



d = {'A':['a','b','c','d'],'B':['m','m','n','n'],'C':[1,2,3,6]}

dp = pd.DataFrame(d)

ds = spark.createDataFrame(dp)

dp['rank'] = dp.groupby('B')['C'].rank('dense',ascending=False)

from pyspark.sql.window import Window

w = Window.partitionBy('B').orderBy(ds.C.desc())

ds = ds.withColumn('rank',F.rank().over(w))

rank vs dense\_rank

d ={'Id':[1,2,3,4,5,6], 'Score': [4.00, 4.00, 3.85, 3.65, 3.65, 3.50]}

data = pd.DataFrame(d)

dp = data.copy()

ds = spark.createDataFrame(data)

dp['Rank\_dense'] = dp['Score'].rank(method='dense',ascending =False)

dp['Rank'] = dp['Score'].rank(method='min',ascending =False)

dp

import pyspark.sql.functions as F

from pyspark.sql.window import Window

w = Window.orderBy(ds.Score.desc())

ds = ds.withColumn('Rank\_spark\_dense',F.dense\_rank().over(w))

ds = ds.withColumn('Rank\_spark',F.rank().over(w))

ds.show()

**Colab :** [**Link**](https://colab.research.google.com/drive/1S5LB2yWn23NmwQr8-Lg39QVWUQHQuZSw?usp=sharing)