So far, we looked into SQL, R and Python and this post will be about Python Koalas package. A special implementation of pandas DataFrame API on Apache Spark. Data Engineers and data scientist love Python pandas, since it makes data preparation with pandas easier, faster and more productive. And Koalas is a direct “response” to make writing and coding on Spark, easier and more familiar. Also follow the official documentation with full description of the package.

Koalas come pre-installed on Databricks Runtine 7.1 and above and we can use package directly in the Azure Databricks notebook. Let us check the Runtime version. Launch your Azure Databricks environment, go to clusters and there you should see the version:

My cluster is rocking Databricks Runtime 7.3. So create a new notebook and name it: *Day13_Py_Koalas* and select the Language: *Python*. And attach the notebook to your cluster.

1.Object Creation

Before going into sample Python code, we must import the following packages: pandas and numpy so we can create from or convert from/to Databricks Koalas.

```
import databricks.koalas as ks
import pandas as pd
import numpy as np
```

Creating a Koalas Series by passing a list of values, letting Koalas create a default integer index:

```
s = ks.Series([1, 3, 5, np.nan, 6, 8])
```

Creating a Koalas DataFrame by passing a dict of objects that can be converted to series-like.

```
kdf = ks.DataFrame(
    {'a': [1, 2, 3, 4, 5, 6],
     'b': [100, 200, 300, 400, 500, 600],
     'c': ["one", "two", "three", "four", "five", "six"]},
    index=[10, 20, 30, 40, 50, 60])
```

with the result:

```
1 kdf
```

▸ (1) Spark Jobs

Out[61]:

	a	b	c
20	2	200	two
50	5	500	five
10	1	100	one
30	3	300	three
40	4	400	four
60	6	600	six

Command took 0.23 seconds -- by tomaz.kastrun@gmail.com

Now, let's create a pandas DataFrame by passing a numpy array, with a datetime index and labeled columns:

```
dates = pd.date_range('20200807', periods=6)
pdf = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))
```

and getting the results as pandas dataframe:

```
1 pdf
```

Out[65]:

	A	B	C	D
2020-08-07	0.583361	0.542618	1.684222	-0.867707
2020-08-08	1.067589	0.624555	-0.068942	1.344479
2020-08-09	-0.552111	-1.037478	0.594816	-1.351690
2020-08-10	0.102291	-0.022990	-1.241653	-0.378568
2020-08-11	-0.227934	0.021743	0.034619	-0.041365
2020-08-12	0.555649	0.704804	1.352279	-2.645881

Command took 0.07 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020, 16:58

Pandas dataframe can easly be converted to Koalas dataframe:

```
kdf = ks.from_pandas(pdf)
type(kdf)
```

With type of: *Out[67]: databricks.koalas.frame.DataFrame*

And we can output the dataframe to get the same result as with pandas dataframe:

```
kdf
```

```
1 kdf
```

▸ (1) Spark Jobs

Out[68]:

	A	B	C	D
2020-08-07	0.583361	0.542618	1.684222	-0.867707
2020-08-09	-0.552111	-1.037478	0.594816	-1.351690
2020-08-10	0.102291	-0.022990	-1.241653	-0.378568
2020-08-12	0.555649	0.704804	1.352279	-2.645881
2020-08-08	1.067589	0.624555	-0.068942	1.344479
2020-08-11	-0.227934	0.021743	0.034619	-0.041365

Command took 0.24 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020, 16:58:

Also, it is possible to create a Koalas DataFrame from Spark DataFrame. We need to load additional pyspark package first, then create a SparkSession and create a Spark Dataframe.

```
#Load package
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
sdf = spark.createDataFrame(pdf)
```

Since spark is lazy we need to explicitly call the show function in order to see the spark dataframe.

```
sdf.show()
```

```
1 sdf.show()
```

▸ (1) Spark Jobs

```
+--------------------+--------------------+--------------------+--------------------+
|                   A|                   B|                   C|                   D|
+--------------------+--------------------+--------------------+--------------------+
| -1.1689549603641773|  0.7588652476167718|  1.7459969435539628|   0.845267896374577|
| -0.3923045401153845|  0.5086411203060197|-0.11787383907584141| 0.07510514651050416|
| -0.3425318608882894|  -2.035189349859852|  -1.817620326316789|  1.3813570086265254|
|  1.7771847598360662|-0.17698317337089955|  1.0373693533342232|  0.6462652119385413|
|-0.14557249096743005| 0.81917569558099388|-0.34679748225680733|  0.3344519324746689|
|   0.539892399712819|  0.1573278558945495| -0.5348724510873413|  -0.719043357398834|
|0.824282063754262298| -1.5312536427531263|  1.1881621496839996|  0.2298607936363347|
|  -1.108792347206274| -1.1751415776415328|  0.3061500844016105|  1.0104386820455036|
|  1.8818756229022195|  -1.078588339799867|  0.7004485860842163|0.023512739813084983|
|  0.2687568964797834| 0.13617593771084893|-0.03640706023098078|  -1.340351008623156|
|-0.44554910638672245|  1.4176558266443475|  0.4618036848788153|  1.2698683188492514|
|  1.1658900230759883| -1.7216877132012105| -1.4657419778414131|  0.5833060685701208|
|-0.18454750123495928| -0.5666327735989798| -0.5118053290396264|  0.3196865714675734|
|  1.9709480502455703|-0.05201957510144157|  1.0850428095122622| 0.12695484743782062|
| -1.1823818661785088| -2.2355978650808783|  1.4616618143640576|  0.2894882015736548|
| 0.13994698907257003| -0.7177851025890001|  0.2925888516528143|   1.576700680186025|
| -0.8321144992916357|  0.5777443201191999| -0.9898282954575567|  0.7482598338988605|
| -2.6058763688454234|-0.28865812329256907|  0.7341661089432205|-0.27367320943249174|
+--------------------+--------------------+--------------------+--------------------+
```

Command took 0.09 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020, 17:00:23 on databbricks_cl1_Standard

Creating Koalas DataFrame from Spark DataFrame. `to_koalas()` is automatically attached to Spark DataFrame and available as an API when Koalas is imported.

```
kdf = sdf.to_koalas()
```

```
1 kdf
```

▸ (2) Spark Jobs

Out[117]:

	A	B	C	D
0	-1.168955	0.750865	1.745997	0.845268
1	-0.392305	0.508641	-0.117874	0.075105
2	-0.342532	-2.035189	-1.817620	1.301357
3	1.777185	-0.176983	1.037369	0.646265
4	-0.145572	0.019176	-0.346797	0.334452
5	0.539892	0.157328	-0.534872	-0.719043
6	0.024282	-1.531254	1.188162	0.229861
7	-1.108792	-1.175142	0.306150	1.010439
8	1.881876	-1.078588	0.700449	0.023513
9	0.268757	0.136176	-0.036407	-1.340351
10	-0.445549	1.417656	0.461804	1.269868
11	1.165890	-1.721688	-1.465742	0.583306

2. Viewing data

See the top rows of the frame. The results may not be the same as pandas though: unlike pandas, the data in a Spark dataframe is not *ordered*, it has no intrinsic notion of index. When asked for the head of a dataframe, Spark will just take the requested number of rows from a partition.

```
kdf.head()
```

```
1 kdf.head()
```

▸ (2) Spark Jobs

Out[119]:

	A	B	C	D
0	-1.168955	0.750865	1.745997	0.845268
1	-0.392305	0.508641	-0.117874	0.075105
2	-0.342532	-2.035189	-1.817620	1.301357
3	1.777185	-0.176983	1.037369	0.646265
4	-0.145572	0.019176	-0.346797	0.334452

Command took 0.30 seconds -- by tomaz.kastrun@gmail.c

You can also display the index, columns, and the underlying numpy data.

```
kdf.index
kdf.columns
kdf.to_numpy()
```

```
1 kdf.index
```

```
Out[120]: Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
             ...
             990, 991, 992, 993, 994, 995, 996, 997, 998, 999],
            dtype='int64', length=1000)
```

Command took 0.05 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020, 17:00:24 on databbricks_cli_Standard

Cmd 37

```
1 kdf.index
```

```
Out[121]: Index(['A', 'B', 'C', 'D'], dtype='object')
```

Command took 0.03 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020, 17:00:24 on databbricks_cli_Standard

Cmd 38

```
1 kdf.to_numpy()
```

▸ (2) Spark Jobs

```
Out[122]: array([[-1.16895496,  0.75086525,  1.74599694,  0.8452679 ],
       [-0.39230454,  0.50864112, -0.11787384,  0.07510515],
       [-0.34253186, -2.03518935, -1.81762033,  1.301357  ],
       ...,
       [ 0.48210446,  0.49055358,  0.71267237,  0.78550583],
       [ 1.77878349, -0.46146581, -2.05455204, -0.96552458],
       [-1.08240801, -0.09045405,  1.71119032,  2.48669479]])
```

Command took 0.23 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020, 17:00:24 on databbricks_cli_Standard

And you can also use describe function to get a statistic summary of your data:

`kdf.describe()`

```
1 kdf.describe()
```

▸ (3) Spark Jobs

Out[123]:

	A	B	C	D
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	-0.009920	-0.004681	0.023628	0.003475
std	1.003006	0.985395	1.040173	1.044633
min	-2.772882	-2.908391	-3.225095	-3.048961
25%	-0.671414	-0.648423	-0.704327	-0.689532
50%	0.014792	-0.002291	-0.002662	0.042912
75%	0.666778	0.655109	0.712672	0.682784
max	3.455867	3.142860	3.968320	3.414681

Command took 0.77 seconds -- by tomaz.kastrun@gmail.com at 13/12/2020,

You can also transpose the data, by adding a T function:

`kdf.T`

and many other functions. Group is also another great way to get summary of your data. Grouping can be done by “chaining” or adding a group by clause. The internal process – when grouping is applied – happens in three steps:

- Splitting data into groups (base on criteria)
- applying the function and
- combining the results back to data structure.

```
kdf.groupby('A').sum()
#or
kdf.groupby(['A', 'B']).sum()
```

Both are grouping data, first time on Column A and second time on both columns A and B:

3. Plotting data

Databricks Koalas is also compatible with matplotlib and inline plotting. We need to load the package:

```
%matplotlib inline
from matplotlib import pyplot as plt
```

And can continue by creating a simple pandas series:

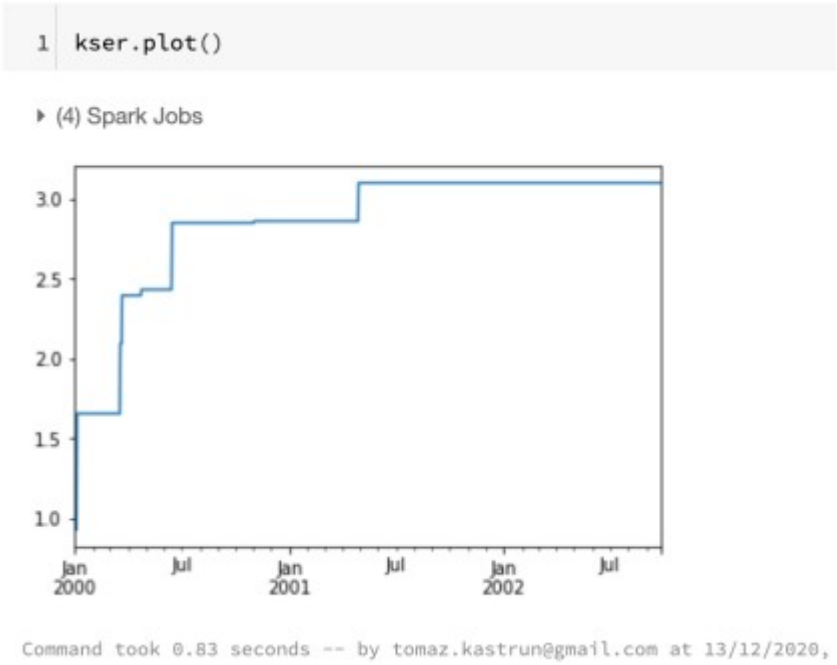
```
pser = pd.Series(np.random.randn(1000),
                 index=pd.date_range('1/1/2000', periods=1000))
```

that can be simply converted to Koalas series:

```
kser = ks.Series(pser)
```

After we have a series in Koalas, we can create cumulative sum of values using series and plot it:

```
kser = kser.cummax()
kser.plot()
```



And many other variations of plot. You can also load the seaborn package, boket package and many others.