march30

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
```

FINISHED

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
```

```
// Load data - adjust the path to the location of your data
val inputPath =  "/Users/skondagadapu/Desktop/data1.csv"
val airTraffic = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath)
```

FINISHED

```
inputPath: String = /Users/skondagadapu/Desktop/data1.csv
airTraffic: org.apache.spark.sql.DataFrame = [date time: timestamp, dewpoint: int ... 2 mor
e fields]
```

```
%pyspark
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
```

FINISHED

```
%pyspark
frame = DataFrame({'data1': np.random.randn(1000), 'data2': np.random.randn(1000)})
factor = pd.cut(frame.data1,4)
factor[:10]
```

FINISHED

```
0     (-0.122, 1.607]
1    (-1.851, -0.122]
2    (-1.851, -0.122]
3    (-1.851, -0.122]
4     (-0.122, 1.607]
5    (-1.851, -0.122]
6     (-0.122, 1.607]
7    (-1.851, -0.122]
8    (-1.851, -0.122]
9     (-0.122, 1.607]
Name: data1, dtype: category
Categories (4, object): [(-3.586, -1.851] < (-1.851, -0.122] < (-0.122, 1.607] < (1.607, 3.
336]]
```

```
%pyspark
def get_stats(group):
  return {'min': group.min(), 'max': group.max(), 'count': group.count(), 'mean': group.me
```

FINISHED

```
%pyspark
grouped = frame.data2.groupby(factor)
grouped.apply(get_stats).unstack()
```

FINISHED

data1	count	max	mean	min
(-3.586, -1.851]	27.0	2.343049	0.186808	-1.742592
(-1.851, -0.122]	418.0	2.646796	-0.021024	-2.856911
(-0.122, 1.607]	497.0	3.136237	0.053193	-3.360784
(1.607, 3.336]	58.0	1.652565	-0.051522	-2.388252

```
%pyspark
grouping = pd.qcut(frame.data1, 10, labels=False)
```

FINISHED

```
%pyspark
grouped = frame.data2.groupby(grouping)
grouped.apply(get_stats).unstack()
```

FINISHED

data1	count	max	mean	min
0	100.0	2.343049	-0.036178	-2.121974
1	100.0	2.200778	0.010417	-2.719506
2	100.0	2.228338	-0.025564	-2.224614
3	100.0	1.878531	0.022785	-2.856911
4	100.0	2.646796	-0.009298	-2.308381
5	100.0	3.136237	0.190603	-2.105550
6	100.0	2.335017	-0.015245	-3.360784
7	100.0	2.494880	0.049510	-3.198931
8	100.0	1.892202	0.038340	-2.379325
9	100.0	2.920185	-0.028326	-2.388252

```
%pyspark
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
```

ERROR

```
paragraph_1490916315579_-1249118938's Interpreter %pyspar not found
```

```
%pyspark
%pyspark
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
```

ERROR

```
Traceback (most recent call last):
  File "/var/folders/jv/jdgs4kr157x42f0cfj4qd_f87f5nz4/T/zeppelin_pyspark-8852624417992631630.py", line 323, in <module>
    code = compile('\n'.join(final_code), '<stdin>', 'exec', ast.PyCF_ONLY_AST, 1)
  File "<stdin>", line 1
    %pyspark
    ^
SyntaxError: invalid syntax
```

```
%pyspark
s = Series(np.random.randn(6))
s[::2] = np.nan
s
s.fillna(s.mean())
```

FINISHED

```
0    0.239830
1   -0.172362
2    0.239830
3    0.715481
4    0.239830
5    0.176371
dtype: float64
```

```
%pyspark
states = ['Ohio', 'New York', 'Vermont', 'Florida', 'Oregon', 'Nevada', 'Califronia', 'Idaho']
group_key = ['East'] * 4 + ['West'] * 4
data = Series(np.random.randn(8), index=states)
data[['Vermont','Nevada','Idaho']] = np.nan
data
```

FINISHED

```
Ohio          0.832069
New York     -1.398191
Vermont            NaN
Florida      -0.407542
Oregon       -0.672250
Nevada             NaN
Califronia    0.889976
Idaho              NaN
dtype: float64
```

```
%pyspark
data.groupby(group_key).mean()
```

FINISHED

```
East   -0.324555
West    0.108863
dtype: float64
```

```
%pyspark
fill_mean = lambda g : g.fillna(g.mean())
data.groupby(group_key).apply(fill_mean)
```

FINISHED

```
Ohio          0.832069
New York     -1.398191
Vermont      -0.324555
Florida      -0.407542
Oregon       -0.672250
Nevada        0.108863
Califronia    0.889976
Idaho         0.108863
dtype: float64
```

```
%pyspark
fill_values = {'East': 0.5, 'West': -1}
fill_func = lambda g: g.fillna(fill_values[g.name])
data.groupby(group_key).apply(fill_func)
```

FINISHED

```
Ohio          0.832069
New York     -1.398191
Vermont       0.500000
Florida      -0.407542
Oregon       -0.672250
Nevada       -1.000000
Califronia    0.889976
Idaho        -1.000000
dtype: float64
```

```
%pyspark
df = DataFrame({'category': ['a','a','a','a','b','b','b','b'], 'data': np.random.randn(8),
df
```

FINISHED

```
  category      data   weights
0        a  0.939235  0.462005
1        a -0.114636  0.778478
2        a  1.490270  0.408720
3        a -0.483165  0.693288
4        b -0.797461  0.148614
5        b  0.866526  0.873340
6        b -0.327098  0.312934
7        b  0.216607  0.982600
```

```
%pyspark
grouped = df.groupby('category')
get_wavg = lambda g: np.average(g['data'], weights=g['weights'])
grouped.apply(get_wavg)
```

FINISHED

```
category
a    0.264172
b    0.323081
dtype: float64
```

```
%pyspark
grouped = df.groupby('category')
get_wavg = lambda g: np.average(g['data'], weights=g['weights'])
grouped.apply(get_wavg)
```

FINISHED

```
category
a    0.264172
b    0.323081
dtype: float64
```

```
%pyspark
```

READY