sophie.gallet_city_revenue_lab

```
%spark2.pyspark
#load file
rdd = sc.wholeTextFiles("city_revenue")
rdd.take(10)
```

SPARK JOBS FINISHED

```
<console>:5: error: object zeppelin is not a member of package org.apach
e
  var value: org.apache.zeppelin.spark.SparkZeppelinContext = _
                         ^
<console>:6: error: object zeppelin is not a member of package org.apach
e
  def set(x: Any) = value = x.asInstanceOf[org.apache.zeppelin.spark.Spa
rkZeppelinContext]
                                                  ^
[(u'hdfs://hdfs-nn-1.au.adaltas.cloud:8020/user/sophie.gallet-dsti/city_
revenue/anger.txt', u'JAN 13\r\nFEB 12\r\nMAR 14\r\nAPR 15\r\nMAY 12\r\n
JUN 15\r\nJUL 19\r\nAUG 15\r\nSEP 13\r\nOCT 8\r\nNOV 14\r\nDEC 16'), (u'
hdfs://hdfs-nn-1.au.adaltas.cloud:8020/user/sophie.gallet-dsti/city_reve
nue/lyon.txt', u'JAN 13\r\nFEB 12\r\nMAR 14\r\nAPR 15\r\nMAY 12\r\nJUN 1
5\r\nJUL 19\r\nAUG 25\r\nSEP 13\r\nOCT 11\r\nNOV 22\r\nDEC 22'), (u'hdfs
://hdfs-nn-1.au.adaltas.cloud:8020/user/sophie.gallet-dsti/city_revenue/
marseilles_1.txt', u'JAN 21\r\nFEB 21\r\nMAR 21\r\nAPR 27\r\nMAY 25\r\nJ
UN 25\r\nJUL 21\r\nAUG 22\r\nSEP 23\r\nOCT 28\r\nNOV 24\r\nDEC 26'), (u'
```

First let's extract the final values we need one by one

READY

```
%spark2.pyspark
rdd_store = rdd.map(lambda v: v[0].split('/')[-1].split('.')[0])
rdd_store.take(10)
```

SPARK JOBS FINISHED

```
[u'anger', u'lyon', u'marseilles_1', u'marseilles_2', u'orlean', u'paris
_2', u'paris_3', u'nantes', u'nice', u'paris_1']
```

```
%spark2.pyspark
rdd_city = rdd.map(lambda v: v[0].split('/')[-1].split('.')[0].split('_'
rdd_city.take(5)
```

SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=4) FINISHED

```
[u'anger', u'lyon', u'marseilles', u'marseilles', u'orlean']
```

```
%spark2.pyspark SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=5) FINISHED
rdd_month = rdd.flatMap(lambda line: line[1].split('\n')) \
    .map(lambda t: t.split(' ')[0])
rdd_month.take(10)
```

```
[u'JAN', u'FEB', u'MAR', u'APR', u'MAY', u'JUN', u'JUL', u'AUG', u'SEP',
u'OCT']
```

```
%spark2.pyspark SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=6) FINISHED
rdd_rev = rdd.flatMap(lambda line: line[1].split('\n')) \
    .map(lambda t: t.split(' ')[1].split('\r')[0])
rdd_rev.take(10)
```

```
[u'13', u'12', u'14', u'15', u'12', u'15', u'19', u'15', u'13', u'8']
```

```
Now let's combine it all together                                READY
```

```
%spark2.pyspark SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=7) FINISHED
# First we break it down between a key and a value - and remove some par
rdd1 = rdd.map(lambda v: (v[0].split("/")[-1], v[1]))
rdd1.take(3)
```

```
[(u'anger.txt', u'JAN 13\r\nFEB 12\r\nMAR 14\r\nAPR 15\r\nMAY 12\r\nJUN
15\r\nJUL 19\r\nAUG 15\r\nSEP 13\r\nOCT 8\r\nNOV 14\r\nDEC 16'), (u'lyon
.txt', u'JAN 13\r\nFEB 12\r\nMAR 14\r\nAPR 15\r\nMAY 12\r\nJUN 15\r\nJUL
19\r\nAUG 25\r\nSEP 13\r\nOCT 11\r\nNOV 22\r\nDEC 22'), (u'marseilles_1.
txt', u'JAN 21\r\nFEB 21\r\nMAR 21\r\nAPR 27\r\nMAY 25\r\nJUN 25\r\nJUL
21\r\nAUG 22\r\nSEP 23\r\nOCT 28\r\nNOV 24\r\nDEC 26')]
```

```
%spark2.pyspark SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=8) FINISHED
# Then we clean the store name even more
rdd2 = rdd1.map(lambda v: (v[0].split(".txt")[0], v[1]))
rdd2.take(3)
```

```
[(u'anger', u'JAN 13\r\nFEB 12\r\nMAR 14\r\nAPR 15\r\nMAY 12\r\nJUN 15\r
\nJUL 19\r\nAUG 15\r\nSEP 13\r\nOCT 8\r\nNOV 14\r\nDEC 16'), (u'lyon', u
'JAN 13\r\nFEB 12\r\nMAR 14\r\nAPR 15\r\nMAY 12\r\nJUN 15\r\nJUL 19\r\nA
UG 25\r\nSEP 13\r\nOCT 11\r\nNOV 22\r\nDEC 22'), (u'marseilles_1', u'JAN
21\r\nFEB 21\r\nMAR 21\r\nAPR 27\r\nMAY 25\r\nJUN 25\r\nJUL 21\r\nAUG 22
\r\nSEP 23\r\nOCT 28\r\nNOV 24\r\nDEC 26')]
```

```
%spark2.pyspark SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=9) FINISHED
# now we use flatMapValues() to map each store with the strings 'MON RE\
rdd3 = rdd2.flatMapValues(lambda v: v.split("\r\n"))
rdd3.take(30)
```

```
[(u'anger', u'JAN 13'), (u'anger', u'FEB 12'), (u'anger', u'MAR 14'), (u
'anger', u'APR 15'), (u'anger', u'MAY 12'), (u'anger', u'JUN 15'), (u'an
ger', u'JUL 19'), (u'anger', u'AUG 15'), (u'anger', u'SEP 13'), (u'anger
', u'OCT 8'), (u'anger', u'NOV 14'), (u'anger', u'DEC 16'), (u'lyon', u'
JAN 13'), (u'lyon', u'FEB 12'), (u'lyon', u'MAR 14'), (u'lyon', u'APR 15
'), (u'lyon', u'MAY 12'), (u'lyon', u'JUN 15'), (u'lyon', u'JUL 19'), (u
'lyon', u'AUG 25'), (u'lyon', u'SEP 13'), (u'lyon', u'OCT 11'), (u'lyon'
, u'NOV 22'), (u'lyon', u'DEC 22'), (u'marseilles_1', u'JAN 21'), (u'mar
seilles_1', u'FEB 21'), (u'marseilles_1', u'MAR 21'), (u'marseilles_1',
u'APR 27'), (u'marseilles_1', u'MAY 25'), (u'marseilles_1', u'JUN 25')]
```

```
%spark2.pyspark SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=10) FINISHED
# Finally we break it down even further to get the format (store, city,
formatted_rdd = rdd3.map(lambda v: (v[0],
                                   v[0].split("_")[0],
                                   v[1].split(" ")[0],
                                   v[1].split(" ")[1]))
formatted_rdd.take(10)
```

```
[(u'anger', u'anger', u'JAN', u'13'), (u'anger', u'anger', u'FEB', u'12'
), (u'anger', u'anger', u'MAR', u'14'), (u'anger', u'anger', u'APR', u'1
5'), (u'anger', u'anger', u'MAY', u'12'), (u'anger', u'anger', u'JUN', u
'15'), (u'anger', u'anger', u'JUL', u'19'), (u'anger', u'anger', u'AUG',
u'15'), (u'anger', u'anger', u'SEP', u'13'), (u'anger', u'anger', u'OCT'
, u'8')]
```

```
1) Format (city, store, month, reverse)
2) Average per month of the shop (all stores combined)
3) Total revenue per city for the year
4) Average per month per city (on this 1 year data)
5) Total revenue per store on the year
```

READY

```
%spark2.pyspark
# Average per month of all the shops
q2 = formatted_rdd.map(lambda v: ("France",
                                  int(v[3]))
                       )
q2 = q2.reduceByKey(lambda v1, v2: v1+v2)
q2 = q2.map(lambda v:(v[0],
                      v[1]/12))
q2.take(12)
```

SPARK JOBS FINISHED

```
[('France', 301)]
```

```
%spark2.pyspark
#Total revenue per city for the year
q3 = formatted_rdd.map(lambda v: (v[1],
                                  int(v[3]))
                    )
q3 = q3.reduceByKey(lambda v1, v2: v1+v2)
q3.take(10)
```

SPARK JOBS FINISHED

```
[(u'troyes', 214), (u'paris', 1568), (u'anger', 166), (u'toulouse', 177)
, (u'lyon', 193), (u'orlean', 196), (u'rennes', 180), (u'nice', 203), (u
'marseilles', 515), (u'nantes', 207)]
```

```
%spark2.pyspark
# Average per month per city (on this 1 year data)
q4 = formatted_rdd.map(lambda v: ((v[1],v[2]),
                                  int(v[3]))
                       )
q4 = q4.reduceByKey(lambda v1, v2: v1+v2)
q4.take(5)
```

SPARK JOB (http://wrk-2.au.adaltas.cloud:46389/jobs/job?id=15) FINISHED

```
[((u'nantes', u'OCT'), 14), ((u'toulouse', u'JUN'), 18), ((u'nantes', u'
SEP'), 13), ((u'anger', u'AUG'), 15), ((u'anger', u'JUN'), 15)]
```

```
%spark2.pyspark                                          SPARK JOBS  FINISHED
# Total revenue per store on the year
## Create tuples of the form: (store, revenue)
q5 = formatted_rdd.map(lambda v: (v[0],
                                  int(v[3]))
                       )
## Reduce by key (store) and sum the values (revenue)
q5 = q5.reduceByKey(lambda v1, v2: v1+v2)
q5.take(10)
```

```
[(u'troyes', 214), (u'lyon', 193), (u'toulouse', 177), (u'marseilles_2',
231), (u'anger', 166), (u'paris_3', 330), (u'paris_1', 596), (u'orlean',
196), (u'marseilles_1', 284), (u'rennes', 180)]
```

```
%spark2.pyspark                                          SPARK JOBS  FINISHED
# For each month, best store (most revenue)
## Create tuples of the form: (MON, (revenue, store))
q6a = formatted_rdd.map(lambda v: (v[2],
                                   (int(v[3]),v[0])
                                   )
                        )
## Reduce by key (month) and pick the max values (revenue)
q6a = q6a.reduceByKey(lambda v1, v2: max(v1, v2))
## Keep only what was asked (mon + best store)
q6a = q6a.map(lambda v: (v[0], v[1][1]))
q6a.take(20)
```

```
[(u'FEB', u'paris_2'), (u'AUG', u'paris_2'), (u'APR', u'paris_1'), (u'JU
N', u'paris_2'), (u'JUL', u'paris_1'), (u'JAN', u'paris_1'), (u'MAY', u'
paris_2'), (u'NOV', u'paris_2'), (u'MAR', u'paris_2'), (u'DEC', u'paris_
1'), (u'OCT', u'paris_1'), (u'SEP', u'paris_2')]
```

```
%spark2.pyspark                                          SPARK JOBS  FINISHED
# For each month, best store (most revenue) - second option (recommendec
## Create tuples of the form: (MON, (revenue, store))
q6b = formatted_rdd.map(lambda v: (v[2],
                                   (v[0], int(v[3]))
                                   )
                        )
```

```
## Reduce by key (month) and pick the max values (revenue)
q6b = q6b.reduceByKey(lambda v1, v2: v1 if v1[1]>v2[1] else v2)
## Keep only what was asked (mon + best store)
q6b = q6b.map(lambda v: (v[0], v[1][0]))
q6b.take(20)
```

```
[(u'FEB', u'paris_2'), (u'AUG', u'paris_2'), (u'APR', u'paris_1'), (u'JU
N', u'paris_2'), (u'JUL', u'paris_1'), (u'JAN', u'paris_1'), (u'MAY', u'
paris_2'), (u'NOV', u'paris_2'), (u'MAR', u'paris_2'), (u'DEC', u'paris_
1'), (u'OCT', u'paris_1'), (u'SEP', u'paris_2')]
```

```
%spark2.pyspark
mon_dict = {"JAN": 1, "FEB":2, "MAR": 3, "APR":4, }
q6sort = q6b.map()
```

ERROR

```
Fail to execute line 2: q6sort = q6b.map()
Traceback (most recent call last):
  File "/data/1/yarn/local/usercache/sophie.gallet-dsti/appcache/applica
tion_1570004749612_0381/container_e19_1570004749612_0381_01_000001/tmp/z
eppelin_pyspark-1992409125731707177.py", line 380, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 2, in <module>
TypeError: map() takes at least 2 arguments (1 given)
```

```
%spark2.pyspark
```

READY