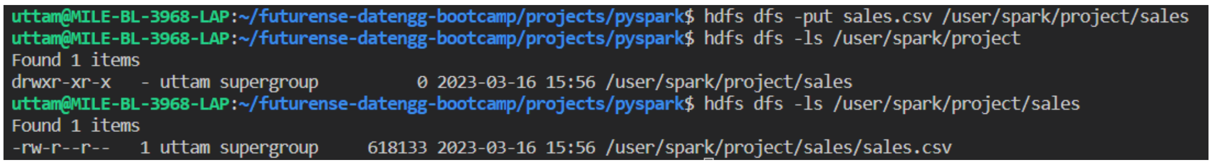
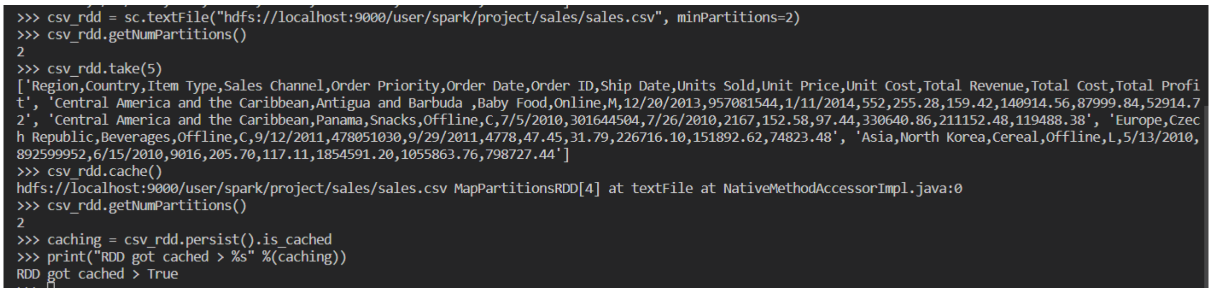
PYSPARK PROJECT:

import the data to hdfs using data ingestion tools



```
uttam@MILE-BL-3968-LAP:~/futurense-datengg-bootcamp/projects/pyspark$ hdfs dfs -put sales.csv /user/spark/project/sales
uttam@MILE-BL-3968-LAP:~/futurense-datengg-bootcamp/projects/pyspark$ hdfs dfs -ls /user/spark/project
Found 1 items
drwxr-xr-x   - uttam supergroup          0 2023-03-16 15:56 /user/spark/project/sales
uttam@MILE-BL-3968-LAP:~/futurense-datengg-bootcamp/projects/pyspark$ hdfs dfs -ls /user/spark/project/sales
Found 1 items
-rw-r--r--   1 uttam supergroup     618133 2023-03-16 15:56 /user/spark/project/sales/sales.csv
```

implement rdd's with pyspark transformations

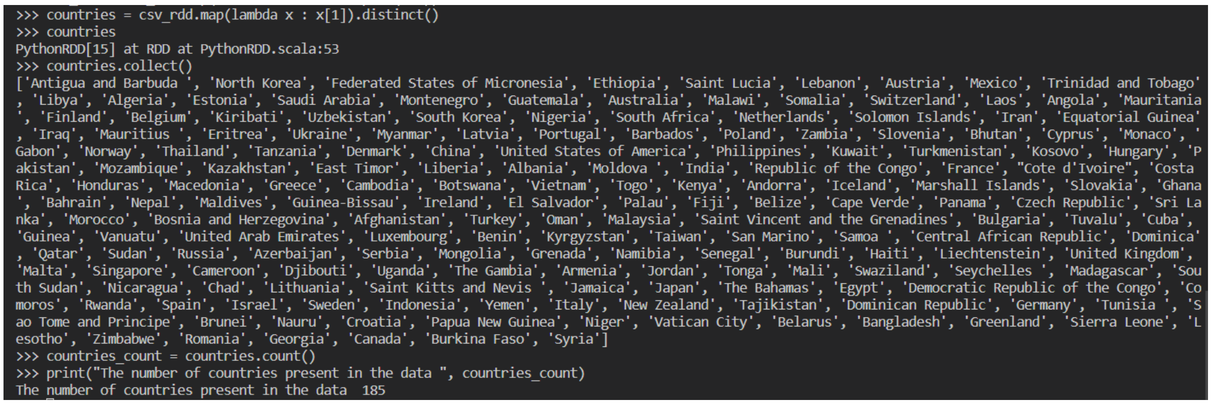


```
>>> csv_rdd = sc.textFile("hdfs://localhost:9000/user/spark/project/sales/sales.csv", minPartitions=2)
>>> csv_rdd.getNumPartitions()
2
>>> csv_rdd.take(5)
['Region,Country,Item Type,Sales Channel,Order Priority,Order Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total Revenue,Total Cost,Total Profi
t', 'Central America and the Caribbean,Antigua and Barbuda ,Baby Food,Online,M,12/20/2013,957081544,1/11/2014,552,255.28,159.42,140914.56,87999.84,52914.7
2', 'Central America and the Caribbean,Panama,Snacks,Offline,C,7/5/2010,301644504,7/26/2010,2167,152.58,97.44,330640.86,211152.48,119488.38', 'Europe,Czec
h Republic,Beverages,Offline,C,9/12/2011,478051030,9/29/2011,4778,47.45,31.79,226716.10,151892.62,74823.48', 'Asia,North Korea,Cereal,Offline,L,5/13/2010,
892599952,6/15/2010,9016,205.70,117.11,1854591.20,1055863.76,798727.44']
>>> csv_rdd.cache()
hdfs://localhost:9000/user/spark/project/sales/sales.csv MapPartitionsRDD[4] at textFile at NativeMethodAccessorImpl.java:0
>>> csv_rdd.getNumPartitions()
2
>>> caching = csv_rdd.persist().is_cached
>>> print("RDD got cached > %s" %(caching))
RDD got cached > True
```

```
from pyspark import SparkContext

# Created Spark Context
sc = SparkContext("local", "Pyspark Batch Processing")

# Loaded CSV file from HDFS to RDD and specified number of partitions as 2
csv_rdd = sc.textFile("hdfs://localhost:9000/user/spark/project/sales/sales.csv")
# csv_rdd.repartition(2)

# Cache the RDD
csv_rdd.cache()
caching = csv_rdd.persist().is_cached
print("RDD got cached > %s" %(caching))

# Formatting the RDD
header = csv_rdd.first()
csv_rdd = csv_rdd.filter(lambda x : x != header)
csv_rdd = csv_rdd.map(lambda line: line.split(","))
```

# 1. Display the number of countries present in the data(Using Pyspark)

```
countries = csv_rdd.map(lambda x : x[1]).distinct()
countries_count = countries.count()
print("The number of countries present in the data ", countries_count)
```

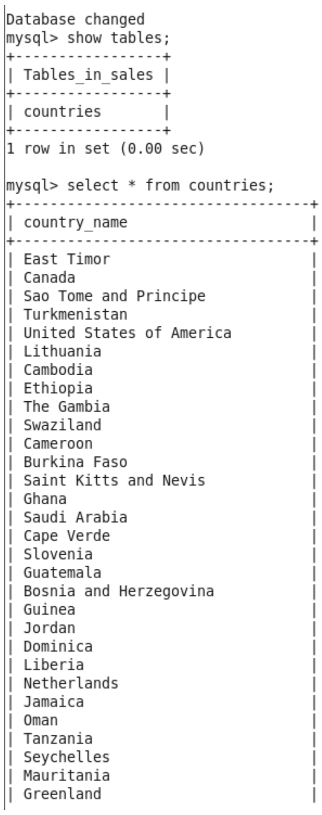


```
>>> countries = csv_rdd.map(lambda x : x[1]).distinct()
>>> countries
PythonRDD[15] at RDD at PythonRDD.scala:53
>>> countries.collect()
['Antigua and Barbuda ', 'North Korea', 'Federated States of Micronesia', 'Ethiopia', 'Saint Lucia', 'Lebanon', 'Austria', 'Mexico', 'Trinidad and Tobago'
, 'Libya', 'Algeria', 'Estonia', 'Saudi Arabia', 'Montenegro', 'Guatemala', 'Australia', 'Malawi', 'Somalia', 'Switzerland', 'Laos', 'Angola', 'Mauritania
', 'Finland', 'Belgium', 'Kiribati', 'Uzbekistan', 'South Korea', 'Nigeria', 'South Africa', 'Netherlands', 'Solomon Islands', 'Iran', 'Equatorial Guinea'
, 'Iraq', 'Mauritius ', 'Eritrea', 'Ukraine', 'Myanmar', 'Latvia', 'Portugal', 'Barbados', 'Poland', 'Zambia', 'Slovenia', 'Bhutan', 'Cyprus', 'Monaco', '
Gabon', 'Norway', 'Thailand', 'Tanzania', 'Denmark', 'China', 'United States of America', 'Philippines', 'Kuwait', 'Turkmenistan', 'Kosovo', 'Hungary', 'P
akistan', 'Mozambique', 'Kazakhstan', 'East Timor', 'Liberia', 'Albania', 'Moldova ', 'India', 'Republic of the Congo', 'France', "Cote d'Ivoire", 'Costa
Rica', 'Honduras', 'Macedonia', 'Greece', 'Cambodia', 'Botswana', 'Vietnam', 'Togo', 'Kenya', 'Andorra', 'Iceland', 'Marshall Islands', 'Slovakia', 'Ghana
', 'Bahrain', 'Nepal', 'Maldives', 'Guinea-Bissau', 'Ireland', 'El Salvador', 'Palau', 'Fiji', 'Belize', 'Cape Verde', 'Panama', 'Czech Republic', 'Sri La
nka', 'Morocco', 'Bosnia and Herzegovina', 'Afghanistan', 'Turkey', 'Oman', 'Malaysia', 'Saint Vincent and the Grenadines', 'Bulgaria', 'Tuvalu', 'Cuba',
'Guinea', 'Vanuatu', 'United Arab Emirates', 'Luxembourg', 'Benin', 'Kyrgyzstan', 'Taiwan', 'San Marino', 'Samoa ', 'Central African Republic', 'Dominica'
, 'Qatar', 'Sudan', 'Russia', 'Azerbaijan', 'Serbia', 'Mongolia', 'Grenada', 'Namibia', 'Senegal', 'Burundi', 'Haiti', 'Liechtenstein', 'United Kingdom',
'Malta', 'Singapore', 'Cameroon', 'Djibouti', 'Uganda', 'The Gambia', 'Armenia', 'Jordan', 'Tonga', 'Mali', 'Swaziland', 'Seychelles ', 'Madagascar', 'Sou
th Sudan', 'Nicaragua', 'Chad', 'Lithuania', 'Saint Kitts and Nevis ', 'Jamaica', 'Japan', 'The Bahamas', 'Egypt', 'Democratic Republic of the Congo', 'Co
moros', 'Rwanda', 'Spain', 'Israel', 'Sweden', 'Indonesia', 'Yemen', 'Italy', 'New Zealand', 'Tajikistan', 'Dominican Republic', 'Germany', 'Tunisia ', 'S
ao Tome and Principe', 'Brunei', 'Nauru', 'Croatia', 'Papua New Guinea', 'Niger', 'Vatican City', 'Belarus', 'Bangladesh', 'Greenland', 'Sierra Leone', 'L
esotho', 'Zimbabwe', 'Romania', 'Georgia', 'Canada', 'Burkina Faso', 'Syria']
>>> countries_count = countries.count()
>>> print("The number of countries present in the data ", countries_count)
The number of countries present in the data  185
```

```
def toCSVLine(data):
   return ','.join(str(d) for d in data)

output1 = countries.coalesce(1)
output1.saveAsTextFile('hdfs://localhost/user/spark/project/output/output1.csv')
```

Sqoop Export Query

```
sqoop export --connect jdbc:mysql://localhost:3306/sales --username root --password
cloudera --table countries --export-dir /user/spark/project/output/output1.csv/part-00000;
```



```
Database changed
mysql> show tables;
+-----------------+
| Tables_in_sales |
+-----------------+
| countries       |
+-----------------+
1 row in set (0.00 sec)

mysql> select * from countries;
+---------------------------------+
| country_name                    |
+---------------------------------+
| East Timor                      |
| Canada                          |
| Sao Tome and Principe           |
| Turkmenistan                    |
| United States of America        |
| Lithuania                       |
| Cambodia                        |
| Ethiopia                        |
| The Gambia                      |
| Swaziland                       |
| Cameroon                        |
| Burkina Faso                    |
| Saint Kitts and Nevis           |
| Ghana                           |
| Saudi Arabia                    |
| Cape Verde                      |
| Slovenia                        |
| Guatemala                       |
| Bosnia and Herzegovina          |
| Guinea                          |
| Jordan                          |
| Dominica                        |
| Liberia                         |
| Netherlands                     |
| Jamaica                         |
| Oman                            |
| Tanzania                        |
| Seychelles                      |
| Mauritania                      |
| Greenland                       |
```

MySQL Query

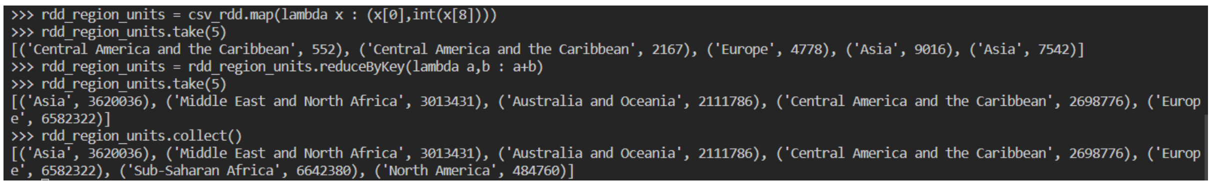
```
CREATE TABLE `countries` (
  `country_name` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL
```

)

Select * from countries;

# 2.Display the number of units sold in each region.(Using Pyspark)

```
rdd_region_units = csv_rdd.map(lambda x : (x[0],int(x[8])))
rdd_region_units = rdd_region_units.reduceByKey(lambda a,b : a+b)
rdd_region_units.collect()
```



```
>>> rdd_region_units = csv_rdd.map(lambda x : (x[0],int(x[8])))
>>> rdd_region_units.take(5)
[('Central America and the Caribbean', 552), ('Central America and the Caribbean', 2167), ('Europe', 4778), ('Asia', 9016), ('Asia', 7542)]
>>> rdd_region_units = rdd_region_units.reduceByKey(lambda a,b : a+b)
>>> rdd_region_units.take(5)
[('Asia', 3620036), ('Middle East and North Africa', 3013431), ('Australia and Oceania', 2111786), ('Central America and the Caribbean', 2698776), ('Europ
e', 6582322)]
>>> rdd_region_units.collect()
[('Asia', 3620036), ('Middle East and North Africa', 3013431), ('Australia and Oceania', 2111786), ('Central America and the Caribbean', 2698776), ('Europ
e', 6582322), ('Sub-Saharan Africa', 6642380), ('North America', 484760)]
```

```
output2 = rdd_region_units.coalesce(1).map(toCSVLine)
output2.saveAsTextFile('hdfs://localhost/user/spark/project/output/output2.csv')
```

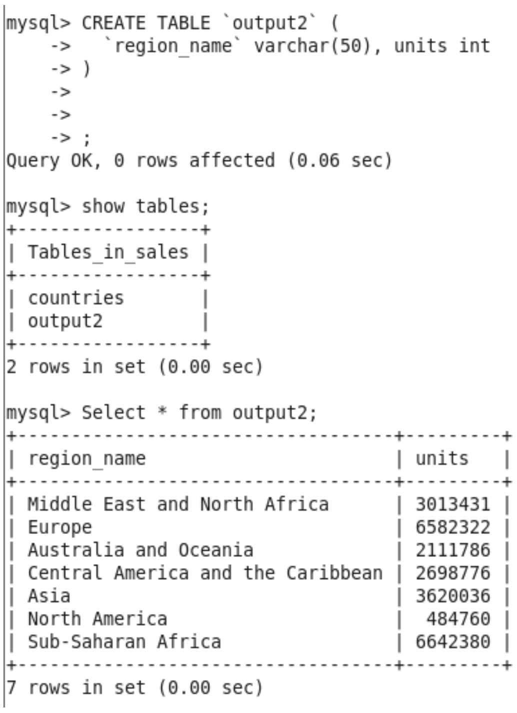
MySQL Query

```
CREATE TABLE `output2` (
  `region_name` varchar(50), units int
)
```

Select * from output2;

Sqoop Export Query

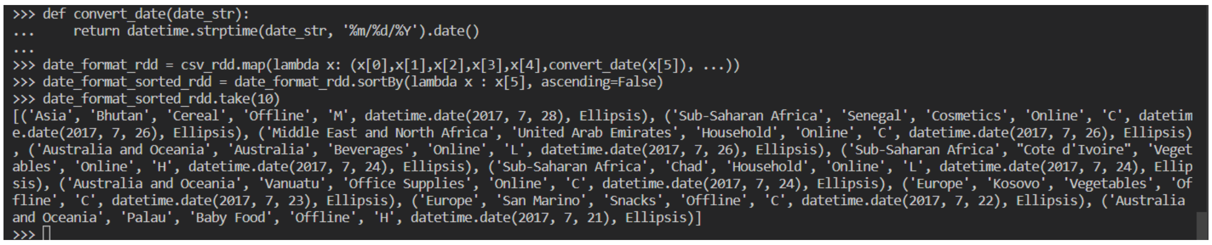
sqoop export --connect jdbc:mysql://localhost:3306/sales --username root --password cloudera --table output2 --export-dir /user/spark/project/output/output2.csv/part-00000;



```
mysql> CREATE TABLE `output2` (
    ->   `region_name` varchar(50), units int
    -> )
    ->
    ->
    -> ;
Query OK, 0 rows affected (0.06 sec)

mysql> show tables;
+-----------------+
| Tables_in_sales |
+-----------------+
| countries       |
| output2         |
+-----------------+
2 rows in set (0.00 sec)

mysql> Select * from output2;
+-----------------------------------+---------+
| region_name                       | units   |
+-----------------------------------+---------+
| Middle East and North Africa      | 3013431 |
| Europe                            | 6582322 |
| Australia and Oceania             | 2111786 |
| Central America and the Caribbean | 2698776 |
| Asia                              | 3620036 |
| North America                     |  484760 |
| Sub-Saharan Africa                | 6642380 |
+-----------------------------------+---------+
7 rows in set (0.00 sec)
```

# 3.Display the 10 most recent sales. (Using Pyspark) '12/20/2013'

```
from datetime import datetime

# Define a function to convert the date column to datetime
def convert_date(date_str):
    return datetime.strptime(date_str, '%m/%d/%Y').date()

date_format_rdd = csv_rdd.map(lambda x: (x[0],x[1],x[2],x[3],x[4],convert_date(x[5])))
date_format_sorted_rdd = date_format_rdd.sortBy(lambda x : x[5], ascending=False)
date_format_sorted_rdd.take(10)
```



```
>>> def convert_date(date_str):
...     return datetime.strptime(date_str, '%m/%d/%Y').date()
...
>>> date_format_rdd = csv_rdd.map(lambda x: (x[0],x[1],x[2],x[3],x[4],convert_date(x[5]), ...))
>>> date_format_sorted_rdd = date_format_rdd.sortBy(lambda x : x[5], ascending=False)
>>> date_format_sorted_rdd.take(10)
[('Asia', 'Bhutan', 'Cereal', 'Offline', 'M', datetime.date(2017, 7, 28), Ellipsis), ('Sub-Saharan Africa', 'Senegal', 'Cosmetics', 'Online', 'C', datetim
e.date(2017, 7, 26), Ellipsis), ('Middle East and North Africa', 'United Arab Emirates', 'Household', 'Online', 'C', datetime.date(2017, 7, 26), Ellipsis)
, ('Australia and Oceania', 'Australia', 'Beverages', 'Online', 'L', datetime.date(2017, 7, 26), Ellipsis), ('Sub-Saharan Africa', "Cote d'Ivoire", 'Veget
ables', 'Online', 'H', datetime.date(2017, 7, 24), Ellipsis), ('Sub-Saharan Africa', 'Chad', 'Household', 'Online', 'L', datetime.date(2017, 7, 24), Ellip
sis), ('Australia and Oceania', 'Vanuatu', 'Office Supplies', 'Online', 'C', datetime.date(2017, 7, 24), Ellipsis), ('Europe', 'Kosovo', 'Vegetables', 'Of
fline', 'C', datetime.date(2017, 7, 23), Ellipsis), ('Europe', 'San Marino', 'Snacks', 'Offline', 'C', datetime.date(2017, 7, 22), Ellipsis), ('Australia
and Oceania', 'Palau', 'Baby Food', 'Offline', 'H', datetime.date(2017, 7, 21), Ellipsis)]
>>> 
```

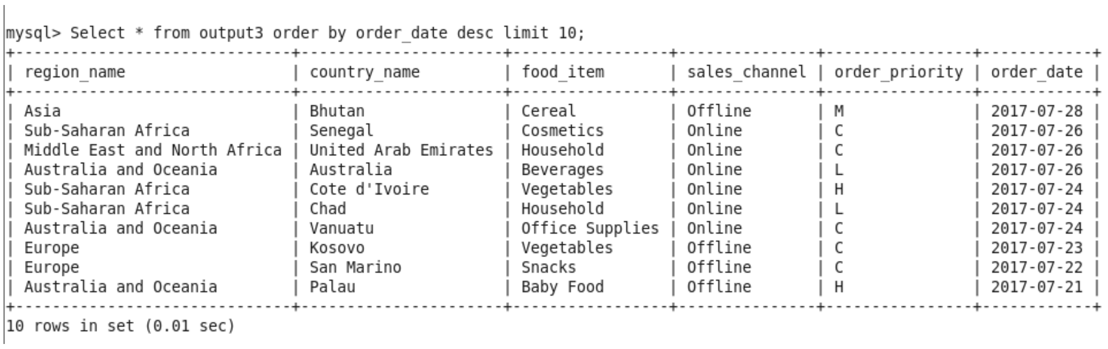
```
output3 = data_format_sorted_rdd.coalesce(1).map(toCSVLine)
output3.saveAsTextFile('hdfs://localhost/user/spark/project/output/output2.csv')
```

MySQL Query

```
CREATE TABLE `output3` (
  region_name varchar(50),
  country_name varchar(50),
  food_item varchar(50),
  sales_channel varchar(50),
  order_priority varchar(1),
  order_date varchar(50)
)

Select * from output3;
```

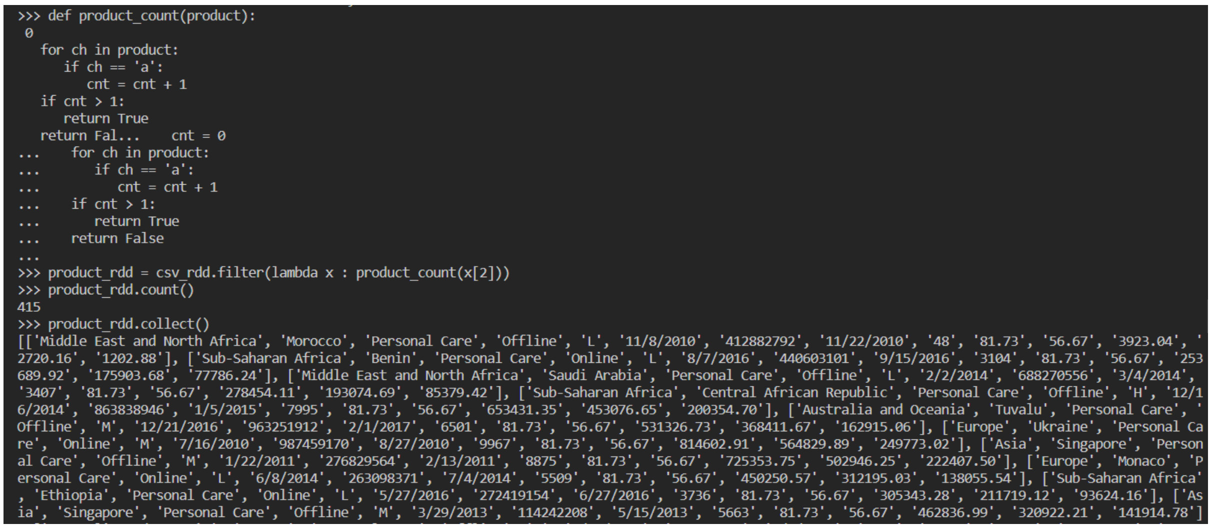
Sqoop Export Query

```
sqoop export --connect jdbc:mysql://localhost:3306/sales --username root --password cloudera --table output3 --export-dir /user/spark/project/output/output3.csv/part-00000;
```



```
mysql> Select * from output3 order by order_date desc limit 10;
+------------------------------+----------------------+-----------------+---------------+----------------+------------+
| region_name                  | country_name         | food_item       | sales_channel | order_priority | order_date |
+------------------------------+----------------------+-----------------+---------------+----------------+------------+
| Asia                         | Bhutan               | Cereal          | Offline       | M              | 2017-07-28 |
| Sub-Saharan Africa           | Senegal              | Cosmetics       | Online        | C              | 2017-07-26 |
| Middle East and North Africa | United Arab Emirates | Household       | Online        | C              | 2017-07-26 |
| Australia and Oceania        | Australia            | Beverages       | Online        | L              | 2017-07-26 |
| Sub-Saharan Africa           | Cote d'Ivoire        | Vegetables      | Online        | H              | 2017-07-24 |
| Sub-Saharan Africa           | Chad                 | Household       | Online        | L              | 2017-07-24 |
| Australia and Oceania        | Vanuatu              | Office Supplies | Online        | C              | 2017-07-24 |
| Europe                       | Kosovo               | Vegetables      | Offline       | C              | 2017-07-23 |
| Europe                       | San Marino           | Snacks          | Offline       | C              | 2017-07-22 |
| Australia and Oceania        | Palau                | Baby Food       | Offline       | H              | 2017-07-21 |
+------------------------------+----------------------+-----------------+---------------+----------------+------------+
10 rows in set (0.01 sec)
```

4.Display the products with atleast 2 occurences of 'a' (Using spark)

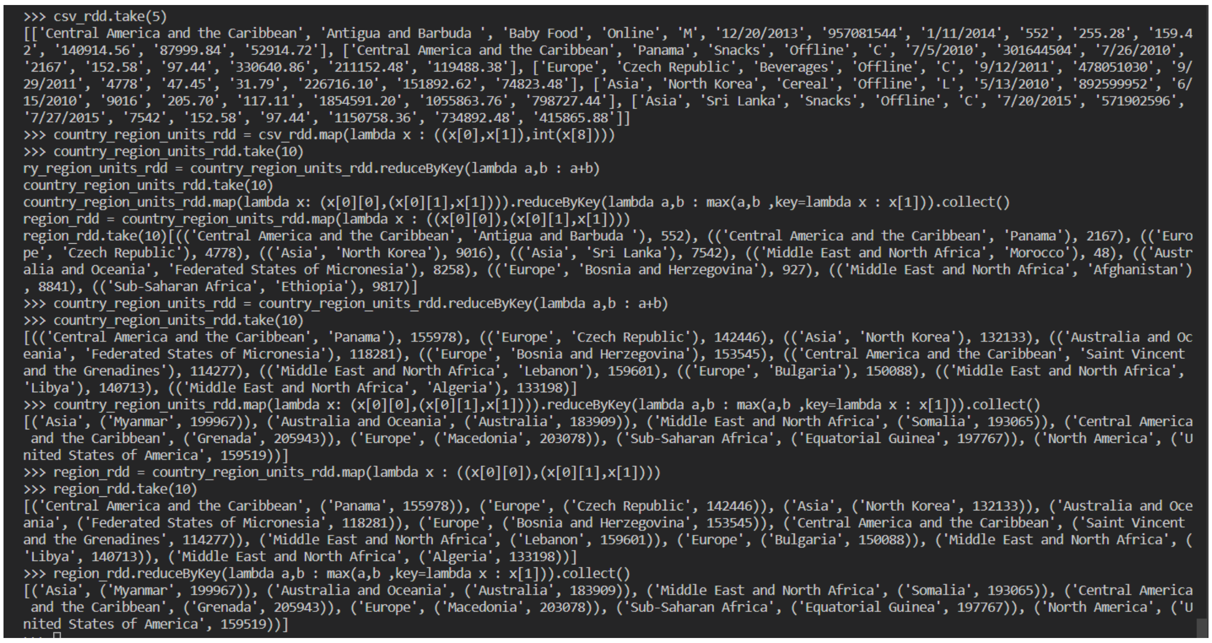


```
>>> def product_count(product):
 0
   for ch in product:
      if ch == 'a':
         cnt = cnt + 1
   if cnt > 1:
      return True
   return Fal...    cnt = 0
...    for ch in product:
...       if ch == 'a':
...          cnt = cnt + 1
...    if cnt > 1:
...       return True
...    return False
...
>>> product_rdd = csv_rdd.filter(lambda x : product_count(x[2]))
>>> product_rdd.count()
415
>>> product_rdd.collect()
[['Middle East and North Africa', 'Morocco', 'Personal Care', 'Offline', 'L', '11/8/2010', '412882792', '11/22/2010', '48', '81.73', '56.67', '3923.04', '
2720.16', '1202.88'], ['Sub-Saharan Africa', 'Benin', 'Personal Care', 'Online', 'L', '8/7/2016', '440603101', '9/15/2016', '3104', '81.73', '56.67', '253
689.92', '175903.68', '77786.24'], ['Middle East and North Africa', 'Saudi Arabia', 'Personal Care', 'Offline', 'L', '2/2/2014', '688270556', '3/4/2014',
'3407', '81.73', '56.67', '278454.11', '193074.69', '85379.42'], ['Sub-Saharan Africa', 'Central African Republic', 'Personal Care', 'Offline', 'H', '12/1
6/2014', '863838946', '1/5/2015', '7995', '81.73', '56.67', '653431.35', '453076.65', '200354.70'], ['Australia and Oceania', 'Tuvalu', 'Personal Care', '
Offline', 'M', '12/21/2016', '963251912', '2/1/2017', '6501', '81.73', '56.67', '531326.73', '368411.67', '162915.06'], ['Europe', 'Ukraine', 'Personal Ca
re', 'Online', 'M', '7/16/2010', '987459170', '8/27/2010', '9967', '81.73', '56.67', '814602.91', '564829.89', '249773.02'], ['Asia', 'Singapore', 'Person
al Care', 'Offline', 'M', '1/22/2011', '276829564', '2/13/2011', '8875', '81.73', '56.67', '725353.75', '502946.25', '222407.50'], ['Europe', 'Monaco', 'P
ersonal Care', 'Online', 'L', '6/8/2014', '263098371', '7/4/2014', '5509', '81.73', '56.67', '450250.57', '312195.03', '138055.54'], ['Sub-Saharan Africa'
, 'Ethiopia', 'Personal Care', 'Online', 'L', '5/27/2016', '272419154', '6/27/2016', '3736', '81.73', '56.67', '305343.28', '211719.12', '93624.16'], ['As
ia', 'Singapore', 'Personal Care', 'Offline', 'M', '3/29/2013', '114242208', '5/15/2013', '5663', '81.73', '56.67', '462836.99', '320922.21', '141914.78']
```

```
def product_count(product):
  cnt = 0
  for ch in product:
    if ch == 'a':
      cnt = cnt + 1
  if cnt > 1:
    return True
  return False

product_rdd = csv_rdd.filter(lambda x : product_count(x[2]))
product_rdd.count()
product_rdd.collect()
```

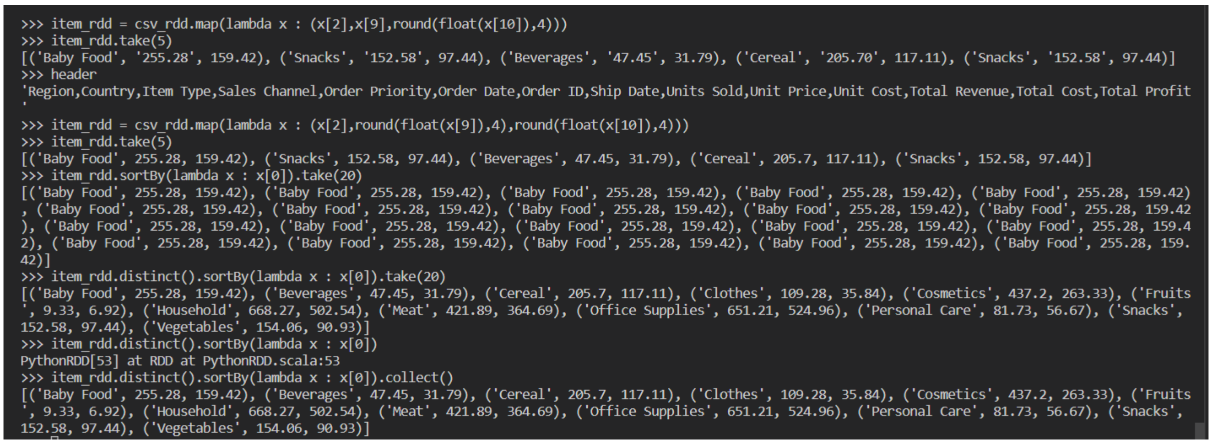
5.Display country in each region with highest units sold. (Using spark)

```
country_region_units_rdd = csv_rdd.map(lambda x : ((x[0],x[1]),int(x[8])))
country_region_units_rdd.take(10)
country_region_units_rdd = country_region_units_rdd.reduceByKey(lambda a,b : a+b)
country_region_units_rdd.take(10)
country_region_units_rdd.map(lambda x: (x[0][0],(x[0][1],x[1]))).reduceByKey(lambda a,b :
max(a,b ,key=lambda x : x[1])).collect()
region_rdd = country_region_units_rdd.map(lambda x : ((x[0][0]),(x[0][1],x[1])))
region_rdd.take(10)
region_rdd.reduceByKey(lambda a,b : max(a,b ,key=lambda x : x[1])).collect()
```



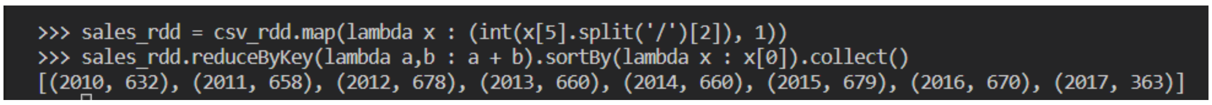
```
>>> csv_rdd.take(5)
[['Central America and the Caribbean', 'Antigua and Barbuda ', 'Baby Food', 'Online', 'M', '12/20/2013', '957081544', '1/11/2014', '552', '255.28', '159.4
2', '140914.56', '87999.84', '52914.72'], ['Central America and the Caribbean', 'Panama', 'Snacks', 'Offline', 'C', '7/5/2010', '301644504', '7/26/2010',
'2167', '152.58', '97.44', '330640.86', '211152.48', '119488.38'], ['Europe', 'Czech Republic', 'Beverages', 'Offline', 'C', '9/12/2011', '478051030', '9/
29/2011', '4778', '47.45', '31.79', '226716.10', '151892.62', '74823.48'], ['Asia', 'North Korea', 'Cereal', 'Offline', 'L', '5/13/2010', '892599952', '6/
15/2010', '9016', '205.70', '117.11', '1854591.20', '1055863.76', '798727.44'], ['Asia', 'Sri Lanka', 'Snacks', 'Offline', 'C', '7/20/2015', '571902596',
'7/27/2015', '7542', '152.58', '97.44', '1150758.36', '734892.48', '415865.88']]
>>> country_region_units_rdd = csv_rdd.map(lambda x : ((x[0],x[1]),int(x[8])))
>>> country_region_units_rdd.take(10)
ry_region_units_rdd = country_region_units_rdd.reduceByKey(lambda a,b : a+b)
country_region_units_rdd.take(10)
country_region_units_rdd.map(lambda x: (x[0][0],(x[0][1],x[1]))).reduceByKey(lambda a,b : max(a,b ,key=lambda x : x[1])).collect()
region_rdd = country_region_units_rdd.map(lambda x : ((x[0][0]),(x[0][1],x[1])))
region_rdd.take(10)[(('Central America and the Caribbean', 'Antigua and Barbuda '), 552), (('Central America and the Caribbean', 'Panama'), 2167), (('Euro
pe', 'Czech Republic'), 4778), (('Asia', 'North Korea'), 9016), (('Asia', 'Sri Lanka'), 7542), (('Middle East and North Africa', 'Morocco'), 48), (('Austr
alia and Oceania', 'Federated States of Micronesia'), 8258), (('Europe', 'Bosnia and Herzegovina'), 927), (('Middle East and North Africa', 'Afghanistan')
, 8841), (('Sub-Saharan Africa', 'Ethiopia'), 9817)]
>>> country_region_units_rdd = country_region_units_rdd.reduceByKey(lambda a,b : a+b)
>>> country_region_units_rdd.take(10)
[(('Central America and the Caribbean', 'Panama'), 155978), (('Europe', 'Czech Republic'), 142446), (('Asia', 'North Korea'), 132133), (('Australia and Oc
eania', 'Federated States of Micronesia'), 118281), (('Europe', 'Bosnia and Herzegovina'), 153545), (('Central America and the Caribbean', 'Saint Vincent
and the Grenadines'), 114277), (('Middle East and North Africa', 'Lebanon'), 159601), (('Europe', 'Bulgaria'), 150088), (('Middle East and North Africa',
'Libya'), 140713), (('Middle East and North Africa', 'Algeria'), 133198)]
>>> country_region_units_rdd.map(lambda x: (x[0][0],(x[0][1],x[1]))).reduceByKey(lambda a,b : max(a,b ,key=lambda x : x[1])).collect()
[('Asia', ('Myanmar', 199967)), ('Australia and Oceania', ('Australia', 183909)), ('Middle East and North Africa', ('Somalia', 193065)), ('Central America
 and the Caribbean', ('Grenada', 205943)), ('Europe', ('Macedonia', 203078)), ('Sub-Saharan Africa', ('Equatorial Guinea', 197767)), ('North America', ('U
nited States of America', 159519))]
>>> region_rdd = country_region_units_rdd.map(lambda x : ((x[0][0]),(x[0][1],x[1])))
>>> region_rdd.take(10)
[('Central America and the Caribbean', ('Panama', 155978)), ('Europe', ('Czech Republic', 142446)), ('Asia', ('North Korea', 132133)), ('Australia and Oce
ania', ('Federated States of Micronesia', 118281)), ('Europe', ('Bosnia and Herzegovina', 153545)), ('Central America and the Caribbean', ('Saint Vincent
and the Grenadines', 114277)), ('Middle East and North Africa', ('Lebanon', 159601)), ('Europe', ('Bulgaria', 150088)), ('Middle East and North Africa', (
'Libya', 140713)), ('Middle East and North Africa', ('Algeria', 133198))]
>>> region_rdd.reduceByKey(lambda a,b : max(a,b ,key=lambda x : x[1])).collect()
[('Asia', ('Myanmar', 199967)), ('Australia and Oceania', ('Australia', 183909)), ('Middle East and North Africa', ('Somalia', 193065)), ('Central America
 and the Caribbean', ('Grenada', 205943)), ('Europe', ('Macedonia', 203078)), ('Sub-Saharan Africa', ('Equatorial Guinea', 197767)), ('North America', ('U
nited States of America', 159519))]
```

6.Display the unit price and unit cost of each item in ascending order. (Using spark)
item_rdd = csv_rdd.map(lambda x : (x[2],round(float(x[9]),4),round(float(x[10]),4)))
item_rdd.distinct().sortBy(lambda x : x[0]).collect()



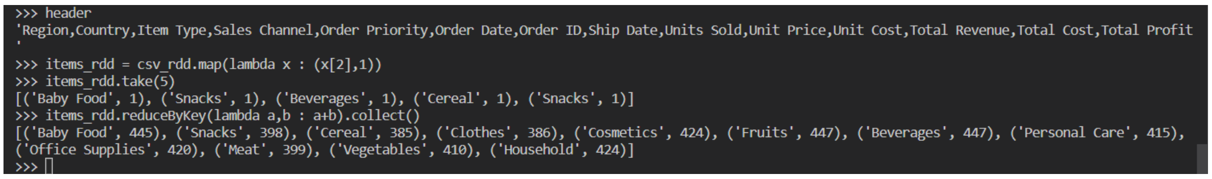
```
>>> item_rdd = csv_rdd.map(lambda x : (x[2],x[9],round(float(x[10]),4)))
>>> item_rdd.take(5)
[('Baby Food', '255.28', 159.42), ('Snacks', '152.58', 97.44), ('Beverages', '47.45', 31.79), ('Cereal', '205.70', 117.11), ('Snacks', '152.58', 97.44)]
>>> header
'Region,Country,Item Type,Sales Channel,Order Priority,Order Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total Revenue,Total Cost,Total Profit
'
>>> item_rdd = csv_rdd.map(lambda x : (x[2],round(float(x[9]),4),round(float(x[10]),4)))
>>> item_rdd.take(5)
[('Baby Food', 255.28, 159.42), ('Snacks', 152.58, 97.44), ('Beverages', 47.45, 31.79), ('Cereal', 205.7, 117.11), ('Snacks', 152.58, 97.44)]
>>> item_rdd.sortBy(lambda x : x[0]).take(20)
[('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42)
, ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42
), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.4
2), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.42), ('Baby Food', 255.28, 159.
42)]
>>> item_rdd.distinct().sortBy(lambda x : x[0]).take(20)
[('Baby Food', 255.28, 159.42), ('Beverages', 47.45, 31.79), ('Cereal', 205.7, 117.11), ('Clothes', 109.28, 35.84), ('Cosmetics', 437.2, 263.33), ('Fruits
', 9.33, 6.92), ('Household', 668.27, 502.54), ('Meat', 421.89, 364.69), ('Office Supplies', 651.21, 524.96), ('Personal Care', 81.73, 56.67), ('Snacks',
152.58, 97.44), ('Vegetables', 154.06, 90.93)]
>>> item_rdd.distinct().sortBy(lambda x : x[0])
PythonRDD[53] at RDD at PythonRDD.scala:53
>>> item_rdd.distinct().sortBy(lambda x : x[0]).collect()
[('Baby Food', 255.28, 159.42), ('Beverages', 47.45, 31.79), ('Cereal', 205.7, 117.11), ('Clothes', 109.28, 35.84), ('Cosmetics', 437.2, 263.33), ('Fruits
', 9.33, 6.92), ('Household', 668.27, 502.54), ('Meat', 421.89, 364.69), ('Office Supplies', 651.21, 524.96), ('Personal Care', 81.73, 56.67), ('Snacks',
152.58, 97.44), ('Vegetables', 154.06, 90.93)]
```

7.Display the number of sales year wise. (Using pyspark)
sales_rdd = csv_rdd.map(lambda x : (int(x[5].split('/')[2]), 1))
sales_rdd.reduceByKey(lambda a,b : a + b).sortBy(lambda x : x[0]).collect()



```
>>> sales_rdd = csv_rdd.map(lambda x : (int(x[5].split('/')[2]), 1))
>>> sales_rdd.reduceByKey(lambda a,b : a + b).sortBy(lambda x : x[0]).collect()
[(2010, 632), (2011, 658), (2012, 678), (2013, 660), (2014, 660), (2015, 679), (2016, 670), (2017, 363)]
```

8.Display the number of orders for each item. (Using pyspark)
items_rdd = csv_rdd.map(lambda x : (x[2],1))
items_rdd.reduceByKey(lambda a,b : a+b).collect()



```
>>> header
'Region,Country,Item Type,Sales Channel,Order Priority,Order Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total Revenue,Total Cost,Total Profit
'
>>> items_rdd = csv_rdd.map(lambda x : (x[2],1))
>>> items_rdd.take(5)
[('Baby Food', 1), ('Snacks', 1), ('Beverages', 1), ('Cereal', 1), ('Snacks', 1)]
>>> items_rdd.reduceByKey(lambda a,b : a+b).collect()
[('Baby Food', 445), ('Snacks', 398), ('Cereal', 385), ('Clothes', 386), ('Cosmetics', 424), ('Fruits', 447), ('Beverages', 447), ('Personal Care', 415),
('Office Supplies', 420), ('Meat', 399), ('Vegetables', 410), ('Household', 424)]
>>> 
```