Copy File:

Copy Files from HDFS to Local file system

hadoop fs -get sam1.txt /home/training/Desktop

Copy Files from Local file system to HDFS

hadoop fs -put /home/training/Desktop/ sam1.txt   sam1.txt

Parallel Copy:

hadoop distcp poems.txt poems1.txt

Merge Files:

hadoop fs -getmerge /user/training/shakespeare   /home/training/Desktop

! For$file"based$RDDS,useSparkContext.textFile$$

– Accepts"a"single"file,"a"wildcard"list"of"files,"or"a"commaUseparated "list"of" files"

– Examples"
– **sc.textFile("myfile.txt")**
– **sc.textFile("mydata/*.log")**
– **sc.textFile("myfile1.txt,myfile2.txt")**
– Each"line"in"the"file(s)"is"a"separate"record"in"the"RDD"

Filesarereferencedbyabsoluteorrela.veURI

– Absolute"URI:"**file:/home/training/myfile.txt**
– RelaDve"URI"(uses"default"file"system):"**myfile.txt**

! **Some$common$ac.ons$**

– **count()**"–""return"the"number"of"elements"
– **take(***n***)**"–"return"an"array"of"the"first"*n*" elements"
– **collect()**–"return"an"array"of"all"elements"
– **saveAsTextFile(***file***)$**–"save"to"text"file(s)

```
mydata =
sc.textFile("purplecow.txt")
> mydata.count()
4
> for line in mydata.take(2):
print line
I've never seen a purple cow.
I never hope to see one;
```

```
val mydata =
sc.textFile("purplecow.txt")
> mydata.count()
4
> for (line <- mydata.take(2))
println(line)
I've never seen a purple cow.
I never hope to see one;
```

```
> mydata = sc.textFile("purplecow.txt")
> mydata_uc = mydata.map(lambda line:
line.upper())
> mydata_filt = \
mydata_uc.filter(lambda line: \
line.startswith('I'))
> mydata_filt.count()
```

Scala:

```
val mydata =sc.textFile("purplecow.txt")
mydata.map(s=>s.toUpperCase).take(5).foreach(println)

val mydata=sc.textFile("SampleDataFile.txt").map(line=> line.toUpperCase).filter( line =>
line.startsWith("S")).foreach(println)
```

Python:

mydata=sc.textFile("SampleDataFile.txt")

>>> for line in mydata.take(2) :

... print line

my_data_uf=mydata.map(lambda line :line.upper()).filter(lambda line: line.startswith('S'))

>>> for line in my_data_uf.take(5) :

... print line

Example:"Passing"Anonymous"FuncDons"

Python:

> mydata.map(lambda line: line.upper()).take(2)

```
>>> from __future__ import print_function
>>> wc.foreach(print)
```

Scala:

> mydata.map(line => line.toUpperCase()).take(2)
> mydata.map(_.toUpperCase()).take(2)

Paird RDD:

Python:

```
users = sc.textFile(file) \
.map(lambda line: line.split('\t')) \
.map(lambda fields: (fields[0],fields[1]))
```

Scala:

```
val users = sc.textFile(file) \
.map(line => line.split('\t')) \
.map(fields => (fields(0),fields(1)))
```

```
val mydata=sc.textFile("sarv.txt").map(line =>
line.split('\t')).map(fields => (fields(0)
,fields(0))).first()
```

```
user001 Fred Flintstone
user090 Bugs Bunny
user111 Harry Potter
```

```
(user001,Fred Flintstone)
(user090,Bugs Bunny)
(user111,Harry Potter)
```

```
sc.textFile(logfile) \
.keyBy(lambda line: line.split(' ')[2])
```

```
> sc.textFile(logfile) \
```

.keyBy(line => line.split(' ')(2))

scala> val mydata=sc.textFile("sarv.txt").keyBy(line=> line.split(' ') (2))

mydata: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[80] at keyBy at <console>:21

scala> mydata.first()

res6: (String, String) = (99788,56.38.234.188 – 99788 "GET /KBDOC-00157.html HTTP/1.0" ...)

read file in spak and convert to uppercase and apply filter:

001,RAJIV,REDDY,21,9848022337,HYDERABAD

002,SIDDARTH,BATTACHARYA,22,9848022338,KOLKATA

003,RAJESH,KHANNA,22,9848022339,DELHI

004,PREETHI,AGARWAL,21,9848022330,PUNE

005,TRUPTHI,MOHANTHY,23,9848022336,BHUWANESHWAR

scala> val mydata = sc.textFile("student_details.txt")

mydata: org.apache.spark.rdd.RDD[String] = student_details.txt MapPartitionsRDD[15] at textFile at <console>:27

scala> val mydata_uc = mydata.map(line => line.toUpperCase)

mydata_uc: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[16] at map at <console>:29

scala> val mydata_filter = mydata_uc.filter(line => line.startsWith("003"))

mydata_filter: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[17] at filter at <console>:31

scala> for (line <- mydata_filter.take(5))

    | println(line)

003,RAJESH,KHANNA,22,9848022339,DELHI

SCALA:

scala> :paste

// Entering paste mode (ctrl-D to finish)

var total =0

for (i <- (1 to 100)){

if(i % 2 ==0)

total +=i

}

scala> 1 +2

res0: Int = 3

scala> println("hi");

hi

```
scala> println("hi")
hi


scala> :paste
// Entering paste mode (ctrl-D to finish)


println("Hello")
1+3


// Exiting paste mode, now interpreting.


Hello
res3: Int = 4


scala> :pase
:pase: no such command.  Type :help for help.


scala> :paste
// Entering paste mode (ctrl-D to finish)


val i=0
i=3
```

```
// Exiting paste mode, now interpreting.

<pastie>:12: error: reassignment to val
i=3
 ^

scala> :paste
// Entering paste mode (ctrl-D to finish)

var i=0
i=4

// Exiting paste mode, now interpreting.

i: Int = 4
i: Int = 4

scala> val i= 1+ 4
i: Int = 5

scala> val i= 1.+(4)
i: Int = 5

scala> val s= "hello Sir"
```

```
s: String = hello Sir


scala> s.
     |
     | toUpperCase
res4: String = HELLO SIR


scala> s.
     | .
<console>:2: error: identifier expected but '.' found.
       .
       ^


scala> s .
     | .
<console>:2: error: identifier expected but '.' found.
       .
       ^


scala> s.
     | .
<console>:2: error: identifier expected but '.' found.
       .
       ^
```

```
scala> s.length

res5: Int = 9


scala> s.to

res6: scala.collection.immutable.IndexedSeq[Char] = Vector(h, e, l, l, o,  , S, i, r)


scala> s.toU

<console>:13: error: value toU is not a member of String

      s.toU

        ^


scala> s.sub

<console>:13: error: value sub is not a member of String

      s.sub

        ^


scala> s.

*          apply          compareTo          dropWhile          format          init          lift
par          replaceAll          size          subSequence   toFloat       toUpperCase

+          applyOrElse          compareToIgnoreCase   endsWith          formatLocal       inits
lines          partition          replaceAllLiterally   slice          substring     toIndexedSeq    toVector

++         canEqual          compose          equals          genericBuilder    intern
linesIterator       patch          replaceFirst          sliding          sum          toInt          transpose

++:        capitalize        concat          equalsIgnoreCase   getBytes          intersect
linesWithSeparators   permutations        repr          sortBy          tail          toIterable     trim
```

+: charAt contains exists getChars isDefinedAt map prefixLength reverse sortWith tails toIterator union

/: chars containsSlice filter groupBy isEmpty matches product reverseIterator sorted take toList unzip

:+ codePointAt contentEquals filterNot grouped isTraversableAgain max r reverseMap span takeRight toLong unzip3

:\ codePointBefore copyToArray find hasDefiniteSize iterator maxBy reduce runWith split takeWhile toLowerCase updated

< codePointCount copyToBuffer flatMap hashCode last min reduceLeft sameElements splitAt to toMap view

<= codePoints corresponds flatten head lastIndexOf minBy reduceLeftOption scan startsWith toArray toSeq withFilter

> collect count fold headOption lastIndexOfSlice mkString reduceOption scanLeft stringPrefix toBoolean toSet zip

>= collectFirst diff foldLeft indexOf lastIndexWhere nonEmpty reduceRight scanRight stripLineEnd toBuffer toShort zipAll

addString combinations distinct foldRight indexOfSlice lastOption offsetByCodePoints reduceRightOption segmentLength stripMargin toByte toStream zipWithIndex

aggregate companion drop forall indexWhere length orElse regionMatches self stripPrefix toCharArray toString

andThen compare dropRight foreach indices lengthCompare padTo replace seq stripSuffix toDouble toTraversable

scala> s.

|

| .

<console>:3: error: identifier expected but '.' found.

.

^

scala> s.

* apply compareTo dropWhile format init lift par replaceAll size subSequence toFloat toUpperCase

+ applyOrElse compareToIgnoreCase endsWith formatLocal inits lines partition replaceAllLiterally slice substring toIndexedSeq toVector

++ canEqual compose equals genericBuilder intern linesIterator patch replaceFirst sliding sum toInt transpose

++: capitalize concat equalsIgnoreCase getBytes intersect linesWithSeparators permutations repr sortBy tail toIterable trim

+: charAt contains exists getChars isDefinedAt map prefixLength reverse sortWith tails toIterator union

/: chars containsSlice filter groupBy isEmpty matches product reverseIterator sorted take toList unzip

:+ codePointAt contentEquals filterNot grouped isTraversableAgain max r reverseMap span takeRight toLong unzip3

:\ codePointBefore copyToArray find hasDefiniteSize iterator maxBy reduce runWith split takeWhile toLowerCase updated

< codePointCount copyToBuffer flatMap hashCode last min reduceLeft sameElements splitAt to toMap view

<= codePoints corresponds flatten head lastIndexOf minBy reduceLeftOption scan startsWith toArray toSeq withFilter

> collect count fold headOption lastIndexOfSlice mkString reduceOption scanLeft stringPrefix toBoolean toSet zip

>= collectFirst diff foldLeft indexOf lastIndexWhere nonEmpty reduceRight scanRight stripLineEnd toBuffer toShort zipAll

addString combinations distinct foldRight indexOfSlice lastOption offsetByCodePoints reduceRightOption segmentLength stripMargin toByte toStream zipWithIndex

aggregate companion drop forall indexWhere length orElse regionMatches self stripPrefix toCharArray toString

andThen compare dropRight foreach indices lengthCompare padTo replace seq stripSuffix toDouble toTraversable

```
scala> s.reverse

res9: String = riS olleh


scala> s.s

sameElements scanLeft segmentLength seq slice sortBy sorted split startsWith stripLineEnd stripPrefix subSequence sum

scan scanRight self size sliding sortWith span splitAt stringPrefix stripMargin stripSuffix substring synchronized


scala> s.ss.substring(5,8)

<console>:13: error: value ss is not a member of String

      s.ss.substring(5,8)

        ^


scala> s.substring(5,8)

res11: String = " Si"


scala> (1 to 100)
```

```
res12: scala.collection.immutable.Range.Inclusive = Range 1 to 100


scala> for (i <- (1 to 100)){
     | print i
     | }
<console>:13: error: missing argument list for method print in object Predef
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `print _` or `print(_)` instead of `print`.
       print i
       ^
<console>:13: warning: postfix operator i should be enabled
by making the implicit value scala.language.postfixOps visible.
This can be achieved by adding the import clause 'import scala.language.postfixOps'
or by setting the compiler option -language:postfixOps.
See the Scaladoc for value scala.language.postfixOps for a discussion
why the feature should be explicitly enabled.
       print i
            ^


scala> for (i <- (1 to 100)){
     | print(i
     | )
     | }
```

123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899100

```
scala> for (i <- (1 to 100)){
     | printlm(i
     | }
<console>:3: error: ')' expected but '}' found.
      }
      ^


scala> for (i <- (1 to 100)){
     | println(i)
     | }
1
2
3
4
5
6
7
8
9
10
11
12
```

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

var tot =0
for (i <- (1 to 100)){
total += i
```

```
// Exiting paste mode, now interpreting.

The pasted code is incomplete!

<pastie>:2: error: Missing closing brace `}' assumed here
var tot =0
^
<pastie>:3: error: expected class or object definition
for (i <- (1 to 100)){
^

scala> :paste
// Entering paste mode (ctrl-D to finish)

var tot =0
for (i <- (1 to 100)){
if(i % 2 ==0)
total +=i
}

// Exiting paste mode, now interpreting.

<pastie>:17: error: not found: value total
total +=i
```

```
^


scala> :paste

// Entering paste mode (ctrl-D to finish)


var total =0

for (i <- (1 to 100)){

if(i % 2 ==0)

total +=i

}


// Exiting paste mode, now interpreting.


total: Int = 2550


scala> :paste

// Entering paste mode (ctrl-D to finish)


var toteven=0

var totodd=0

for (i <- (1 to 100))

{

if (i%2==0)

toteven+=i
```

```
else
totodd+=i
}

// Exiting paste mode, now interpreting.

toteven: Int = 2550
totodd: Int = 2500

scala> :paste
// Entering paste mode (ctrl-D to finish)

val lb=1
val ub= 100
var toteven=0
var totodd=0
while (lb<ub)
{
if (lb%2 == 0) toteven+=lb
else totodd+=lb
}

// Exiting paste mode, now interpreting.
```

```
[training@localhost ~]$ scala
Welcome to Scala 2.12.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_60).
Type in expressions for evaluation. Or try :help.

scala> :paste
// Entering paste mode (ctrl-D to finish)

val lb=1
val ub= 100
var toteven=0
var totodd=0
while (lb<ub)
{
if (lb%2 == 0) toteven+=lb
else totodd+=lb
lb+=1
}

// Exiting paste mode, now interpreting.

<console>:21: error: value += is not a member of Int
  Expression does not convert to assignment because receiver is not assignable.
       lb+=1
         ^
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

var lb =1
val ub= 100
var toteven=0
var totodd=0
while (lb<ub)
{
if (lb%2 == 0) toteven+=lb
else totodd+=lb
lb+=1
}



// Exiting paste mode, now interpreting.

lb: Int = 100
ub: Int = 100
toteven: Int = 2450
totodd: Int = 2500

scala> def sum(lb : Int , ub: Int )= {
```

```
     | var total = 0
     | for (i <- lb to ub)
     | {
     | total += i
     | }
     | total
     | }
sum: (lb: Int, ub: Int)Int


scala> sum(1,3)
res2: Int = 6


scala> def sum(lb : Int , ub: Int )= {
     |      | var total = 0
<console>:2: error: ';' expected but 'var' found.
           | var total = 0
             ^


scala>      | for (i <- lb to ub)
<console>:1: error: ';' expected but 'for' found.
           | for (i <- lb to ub)
             ^


scala> def sum(lb : Int , ub: Int )= {
```

```
     | var total = 0
     | for (i <- lb to ub)
     | total += i
     | total
     | }
sum: (lb: Int, ub: Int)Int

scala> sum(1,3)
res3: Int = 6

scala> sum(1,10)
res4: Int = 55

scala> def cube(i: Int)= i*i*i
cube: (i: Int)Int

scala> cube(3)
res5: Int = 27

scala> def sum(func: Int => Int, lb:Int, ub: Int)={
     |
     |
You typed two blank lines.  Starting a new command.
```

```
scala>


scala> def sum(func: Int => Int, lb:Int, ub: Int)={
     | var total =0
     | for (element <- lb to ub){
     | total += func(element)
     | }
     | total
     | }
sum: (func: Int => Int, lb: Int, ub: Int)Int


scala> def id(i: Int)= i
id: (i: Int)Int


scala> sum(id, 1,10)
res6: Int = 55




scala> def sum(func: Int => Int, lb:Int, ub: Int)={
     |
     |
You typed two blank lines.  Starting a new command.
```

```
scala>


scala> def sum(func: Int => Int, lb:Int, ub: Int)={
     | var total =0
     | for (element <- lb to ub){
     | total += func(element)
     | }
     | total
     | }
sum: (func: Int => Int, lb: Int, ub: Int)Int


scala> def id(i: Int)= i
id: (i: Int)Int


scala> sum(id, 1,10)
res6: Int = 55


scala>


scala> class Order(orderId: Int, orderDate: String, orderCustomerId: Int, orderStatus: String){
     | println("I am inside Order Constructor")
     | }
defined class Order
```

```
scala> val order = new Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order@c0b10ae


scala> class Order(orderId: Int, orderDate: String, orderCustomerId: Int, orderStatus: String){

     |       | println("I am inside Order Constructor")

     | override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"

     | }

defined class Order


scala> val order = new Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> order.orderId

<console>:13: error: value orderId is not a member of Order

       order.orderId

            ^


scala>


scala> order.orderId

<console>:13: error: value orderId is not a member of Order

       order.orderId
```

```
          ^


scala> :javap -p Order
Compiled from "<console>"
public class $line17.$read$$iw$$iw$Order {
  private final int orderId;
  private final java.lang.String orderDate;
  private final int orderCustomerId;
  private final java.lang.String orderStatus;
  public java.lang.String toString();
  public $line17.$read$$iw$$iw$Order(int, java.lang.String, int, java.lang.String);
}


scala> order.orderId
<console>:13: error: value orderId is not a member of Order
       order.orderId
             ^


scala> class Order(val orderId: Int, val orderDate: String, val orderCustomerId: Int, val
orderStatus: String){
     |       | println("I am inside Order Constructor")
     | override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"
     | }
defined class Order
```

```
scala> :javap -p Order
Compiled from "<console>"
public class $line22.$read$$iw$$iw$Order {
  private final int orderId;
  private final java.lang.String orderDate;
  private final int orderCustomerId;
  private final java.lang.String orderStatus;
  public int orderId();
  public java.lang.String orderDate();
  public int orderCustomerId();
  public java.lang.String orderStatus();
  public java.lang.String toString();
  public $line22.$read$$iw$$iw$Order(int, java.lang.String, int, java.lang.String);
}

scala> val order = new Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)

scala> order.orderId
res10: Int = 1

scala> class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var
orderStatus: String){
```

```
    |     | println("I am inside Order Constructor")
    | override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"
    | }
defined class Order

scala> :javap -p Order
Compiled from "<console>"
public class $line25.$read$$iw$$iw$Order {
  private int orderId;
  private java.lang.String orderDate;
  private int orderCustomerId;
  private java.lang.String orderStatus;
  public int orderId();
  public void orderId_$eq(int);
  public java.lang.String orderDate();
  public void orderDate_$eq(java.lang.String);
  public int orderCustomerId();
  public void orderCustomerId_$eq(int);
  public java.lang.String orderStatus();
  public void orderStatus_$eq(java.lang.String);
  public java.lang.String toString();
  public $line25.$read$$iw$$iw$Order(int, java.lang.String, int, java.lang.String);
}
```

```
scala> val order = new Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)

scala> order.orderId
res11: Int = 1

scala> order.orderId=2
order.orderId: Int = 2

scala> order.orderId
res12: Int = 2

scala> order.orderId_=3
<console>:1: error: ';' expected but integer literal found.
       order.orderId_=3
                      ^

scala> order.orderId_=(3)

scala> order.orderId
res14: Int = 3

scala> order.orderId_= 3
```

```
<console>:1: error: ';' expected but integer literal found.
       order.orderId_= 3
                     ^


scala> order.orderId_ = 3
<console>:12: error: value orderId_ is not a member of Order
       order.orderId_ = 3
             ^
<console>:13: error: value orderId_ is not a member of Order
       val $ires1 = order.orderId_
                    ^


scala> order.orderId_ = (3)
<console>:12: error: value orderId_ is not a member of Order
       order.orderId_ = (3)
             ^
<console>:13: error: value orderId_ is not a member of Order
       val $ires2 = order.orderId_
                    ^


scala> order.orderId_=(3)


scala> order.orderId_ = (3)
<console>:12: error: value orderId_ is not a member of Order
```

```
      order.orderId_ = (3)
           ^
<console>:13: error: value orderId_ is not a member of Order
      val $ires3 = order.orderId_
                   ^


scala> order.orderId_=3
<console>:1: error: ';' expected but integer literal found.
      order.orderId_=3
                 ^


scala> order.orderId =(3)
order.orderId: Int = 3


scala> object Helloworld{
     | def main(args: Array[String]): Unit {
     | .
<console>:3: error: illegal start of declaration (possible cause: missing `=' in front of current
method body)
      .
      ^


scala> object Helloworld{
     | def main(args: Array[String]): Unit = {
     | println("Hello World")
```

```
    | }
    | }
defined object Helloworld

scala> Helloworld.main(Array(" "))
Hello World

scala> :paste
// Entering paste mode (ctrl-D to finish)

class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var orderStatus: String){
    | println("I am inside Order Constructor")
override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," + orderStatus + ")"
}
object Order {
def apply(orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String   )
:Order={
new Order(orderId,orderDate,orderCustomerId,orderStatus)
}
}

// Exiting paste mode, now interpreting.
```

```
defined class Order

defined object Order


scala> val order = Order.apply(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> :paste

// Entering paste mode (ctrl-D to finish)


class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var orderStatus:
String){

    | println("I am inside Order Constructor")

override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"

}

object Order {

def apply(orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String   )

:Order={

new Order(orderId,orderDate,orderCustomerId,orderStatus)

def apply(orderId: String,  orderDate: String,  orderCustomerId: String,  orderStatus: String   )

:Order={

new Order(orderId.toInt,orderDate,orderCustomerId.toInt,orderStatus)

}

}
```

```
// Exiting paste mode, now interpreting.


The pasted code is incomplete!


<pastie>:2: error: Missing closing brace `}' assumed here

class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var orderStatus: String){

^


scala> :paste

// Entering paste mode (ctrl-D to finish)


lass Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var orderStatus: String){

    | println("I am inside Order Constructor")

override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," + orderStatus + ")"

}

object Order {

def apply(orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String   )

:Order={

new Order(orderId,orderDate,orderCustomerId,orderStatus)

}

def apply(orderId: String,  orderDate: String,  orderCustomerId: String,  orderStatus: String   )
```

```
:Order={

new Order(orderId.toInt,orderDate,orderCustomerId.toInt,orderStatus)

}

}


// Exiting paste mode, now interpreting.


<pastie>:1: error: illegal start of simple expression

lass Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var orderStatus:
String){

         ^


scala> class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var
orderStatus: String){

     |       | println("I am inside Order Constructor")

     | override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"

     | }

defined class Order

warning: previously defined object Order is not a companion to class Order.

Companions must be defined together; you may wish to use :paste mode for this.


scala> object Order {

     | def apply(orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String   )

     | :Order={
```

```
     | new Order(orderId,orderDate,orderCustomerId,orderStatus)
     | }
     | def apply(orderId: String,  orderDate: String,  orderCustomerId: String,  orderStatus: String
)
     | :Order={
     | new Order(orderId.toInt,orderDate,orderCustomerId.toInt,orderStatus)
     | }
     | }
defined object Order
warning: previously defined class Order is not a companion to object Order.
Companions must be defined together; you may wish to use :paste mode for this.


scala> :paste
// Entering paste mode (ctrl-D to finish)


class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var orderStatus:
String){
     println("I am inside Order Constructor")
override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"
}


object Order {


def apply(orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String   ):Order={
```

```
new Order(orderId,orderDate,orderCustomerId,orderStatus)
                                                                    }


def apply(orderId: String,  orderDate: String,  orderCustomerId: String,  orderStatus:
String   ):Order={
new Order(orderId.toInt,orderDate,orderCustomerId.toInt,orderStatus)
                                                                    }


}


// Exiting paste mode, now interpreting.


defined class Order
defined object Order


scala> val order = Order.apply(1,"2017-11-24 00:00:00:000",100,"COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> val order = Order.apply("1","2017-11-24 00:00:00:000","100","COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> :paste
// Entering paste mode (ctrl-D to finish)
```

```
case class Order( orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String){
     println("I am inside Order Constructor")
override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"
}

// Exiting paste mode, now interpreting.

defined class Order

scala> :java -p order
Compiled from "<console>"
public class $line45.$read$$iw$$iw$ {
  public static $line45.$read$$iw$$iw$ MODULE$;
  private final $line43.$read$$iw$$iw$Order order;
  public static {};
  public $line43.$read$$iw$$iw$Order order();
  public $line45.$read$$iw$$iw$();
}

scala> val order = Order.apply(1,"2017-11-24 00:00:00:000",100,"COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)
```

```
scala> Order.orderId

<console>:14: error: value orderId is not a member of object Order

       Order.orderId

            ^


scala> case class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var
orderStatus: String){

     |       println("I am inside Order Constructor")

     | override def toString = "Order(" + orderId + "," + orderDate + "," + orderCustomerId + "," +
orderStatus + ")"

     | }

defined class Order


scala> val order = Order.apply(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> Order.orderId

<console>:14: error: value orderId is not a member of object Order

       Order.orderId

            ^


scala> val order = Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)
```

```
scala> Order.orderId

<console>:14: error: value orderId is not a member of object Order

      Order.orderId

           ^


scala> :paste

// Entering paste mode (ctrl-D to finish)


case class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var
orderStatus: String){

    |       println("I am inside Order Constructor")

}


// Exiting paste mode, now interpreting.


defined class Order
```

```
scala> val order = Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> Order.orderId

<console>:14: error: value orderId is not a member of object Order

       Order.orderId

            ^


scala> case class Order(var orderId: Int, var orderDate: String, var orderCustomerId: Int, var
orderStatus: String){

     | println("I am inside Order Constructor")

     | }

defined class Order


scala> val order = Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")

I am inside Order Constructor

order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)


scala> Order.orderId

<console>:14: error: value orderId is not a member of object Order

       Order.orderId

            ^
```

```
scala> :java -p order
Compiled from "<console>"
public class $line58.$read$$iw$$iw$ {
  public static $line58.$read$$iw$$iw$ MODULE$;
  private final $line57.$read$$iw$$iw$Order order;
  public static {};
  public $line57.$read$$iw$$iw$Order order();
  public $line58.$read$$iw$$iw$();
}

scala> :javap -p order
Compiled from "<console>"
public class $line58.$read$$iw$$iw$ {
  public static $line58.$read$$iw$$iw$ MODULE$;
  private final $line57.$read$$iw$$iw$Order order;
  public static {};
  public $line57.$read$$iw$$iw$Order order();
  public $line58.$read$$iw$$iw$();
}

scala> :javap -p Order
Compiled from "<console>"
public class $line57.$read$$iw$$iw$Order implements scala.Product,scala.Serializable {
  private int orderId;
```

```
private java.lang.String orderDate;

private int orderCustomerId;

private java.lang.String orderStatus;

public int orderId();

public void orderId_$eq(int);

public java.lang.String orderDate();

public void orderDate_$eq(java.lang.String);

public int orderCustomerId();

public void orderCustomerId_$eq(int);

public java.lang.String orderStatus();

public void orderStatus_$eq(java.lang.String);

public $line57.$read$$iw$$iw$Order copy(int, java.lang.String, int, java.lang.String);

public int copy$default$1();

public java.lang.String copy$default$2();

public int copy$default$3();

public java.lang.String copy$default$4();

public java.lang.String productPrefix();

public int productArity();

public java.lang.Object productElement(int);

public scala.collection.Iterator<java.lang.Object> productIterator();

public boolean canEqual(java.lang.Object);

public int hashCode();

public java.lang.String toString();

public boolean equals(java.lang.Object);
```

```
  public $line57.$read$$iw$$iw$Order(int, java.lang.String, int, java.lang.String);
}

scala> val order = Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)

scala> Order.orderId
<console>:14: error: value orderId is not a member of object Order
       Order.orderId
             ^

scala> Order.
apply   curried   toString   tupled   unapply

scala> case class Order( orderId: Int,  orderDate: String,  orderCustomerId: Int,  orderStatus: String){
     |       println("I am inside Order Constructor")
     |
     | }
defined class Order

scala> val order = Order(1,"2017-11-24 00:00:00:000",100,"COMPLETE")
I am inside Order Constructor
order: Order = Order(1,2017-11-24 00:00:00:000,100,COMPLETE)
```

```
scala> Order.orderId
<console>:14: error: value orderId is not a member of object Order
       Order.orderId
             ^

scala> Order.
apply   curried   toString   tupled   unapply

scala> Order.orderDate
<console>:14: error: value orderDate is not a member of object Order
       Order.orderDate
             ^

scala> Collection
<console>:12: error: not found: value Collection
       Collection
       ^

scala> Array
res26: Array.type = scala.Array$@37841334

scala> Set
res27: scala.collection.immutable.Set.type = scala.collection.immutable.Set$@73992ed7
```

```
scala> Map

res28: scala.collection.immutable.Map.type = scala.collection.immutable.Map$@606d6258


scala> List

res29: scala.collection.immutable.List.type = scala.collection.immutable.List$@7e310a69


scala> val a = Array(1,2,3,4,5)

a: Array[Int] = Array(1, 2, 3, 4, 5)


scala> val b= Set(1,2,3,4,1,5,2)

b: scala.collection.immutable.Set[Int] = Set(5, 1, 2, 3, 4)


scala> val c= Map(1,2,3,1,2,3,4,5)

<console>:11: error: type mismatch;

 found   : Int(1)

 required: (?, ?)

      val c= Map(1,2,3,1,2,3,4,5)

                 ^

<console>:11: error: type mismatch;

 found   : Int(2)

 required: (?, ?)

      val c= Map(1,2,3,1,2,3,4,5)

                   ^
```

```
<console>:11: error: type mismatch;
 found   : Int(3)
 required: (?, ?)
      val c= Map(1,2,3,1,2,3,4,5)
                 ^
<console>:11: error: type mismatch;
 found   : Int(1)
 required: (?, ?)
      val c= Map(1,2,3,1,2,3,4,5)
                   ^
<console>:11: error: type mismatch;
 found   : Int(2)
 required: (?, ?)
      val c= Map(1,2,3,1,2,3,4,5)
                     ^
<console>:11: error: type mismatch;
 found   : Int(3)
 required: (?, ?)
      val c= Map(1,2,3,1,2,3,4,5)
                       ^
<console>:11: error: type mismatch;
 found   : Int(4)
 required: (?, ?)
      val c= Map(1,2,3,1,2,3,4,5)
```

```
                        ^
<console>:11: error: type mismatch;
 found   : Int(5)
 required: (?, ?)
      val c= Map(1,2,3,1,2,3,4,5)
                        ^


scala> val c= Map("Hello" ->1,"world"->2)
c: scala.collection.immutable.Map[String,Int] = Map(Hello -> 1, world -> 2)


scala> val b= Set(1,2,3,4,1,5,2)
b: scala.collection.immutable.Set[Int] = Set(5, 1, 2, 3, 4)


scala> val a = Array(1,2,3,4,5)
a: Array[Int] = Array(1, 2, 3, 4, 5)


scala> val b= Set(1,2,3,4,1,5,2)
b: scala.collection.immutable.Set[Int] = Set(5, 1, 2, 3, 4)


scala> val c= Map("Hello" ->1,"world"->2)
c: scala.collection.immutable.Map[String,Int] = Map(Hello -> 1, world -> 2)


scala> val d=List(1,2,3,4,1,5,2)
d: List[Int] = List(1, 2, 3, 4, 1, 5, 2)
```

```
scala> a.foreach(println)
1
2
3
4
5

scala> b.foreach(println)
5
1
2
3
4

scala> c.foreach(println)
(Hello,1)
(world,2)

scala> d.foreach(println)
1
2
3
4
```

```
1

5

2


scala> a(0)

res34: Int = 1


scala> b(0)

res35: Boolean = false


scala> c(0)

<console>:13: error: type mismatch;

 found   : Int(0)

 required: String

       c(0)

         ^


scala> c("Hello")

res37: Int = 1


scala> d(0)

res38: Int = 1


scala> a.
```

++ combinations elemTag hasDefiniteSize lastIndexOfSlice partition sameElements sum toStream

++: companion endsWith head lastIndexWhere patch scan tail toTraversable

+: compose exists headOption lastOption permutations scanLeft tails toVector

/: contains filter indexOf length prefixLength scanRight take transform

:+ containsSlice filterNot indexOfSlice lengthCompare product segmentLength takeRight transpose

:\ copyToArray find indexWhere lift reduce seq takeWhile union

addString copyToBuffer flatMap indices map reduceLeft size to unzip

aggregate corresponds flatten init max reduceLeftOption slice toArray unzip3

andThen count fold inits maxBy reduceOption sliding toBuffer update

apply deep foldLeft intersect min reduceRight sortBy toIndexedSeq updated

applyOrElse diff foldRight isDefinedAt minBy reduceRightOption sortWith toIterable view

array distinct forall isEmpty mkString repr sorted toIterator withFilter

canEqual drop foreach isTraversableAgain nonEmpty reverse span toList zip

clone dropRight genericBuilder iterator orElse reverseIterator splitAt toMap zipAll

collect dropWhile groupBy last padTo reverseMap startsWith toSeq zipWithIndex

collectFirst elemManifest grouped lastIndexOf par runWith stringPrefix toSet

scala> a.max

res39: Int = 5

scala> a.toSet

res40: scala.collection.immutable.Set[Int] = Set(5, 1, 2, 3, 4)

scala> a

res41: Array[Int] = Array(1, 2, 3, 4, 5)

scala> (1 to 100)

res42: scala.collection.immutable.Range.Inclusive = Range 1 to 100

scala> val l= (1 to 100).toList

l: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)

scala> //sum of all even number

scala> l.foreach(println)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

99

100

```
scala> val l= (1 to 100).toList

l: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)

scala> val f=l.filter(element => element%2==0)

f: List[Int] = List(2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100)

scala> val m=f.mal(rec => rec*rec)

<console>:12: error: value mal is not a member of List[Int]

      val m=f.mal(rec => rec*rec)

            ^

scala> val m=f.map(rec => rec*rec)

m: List[Int] = List(4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784, 900, 1024, 1156, 1296, 1444, 1600, 1764, 1936, 2116, 2304, 2500, 2704, 2916, 3136, 3364, 3600, 3844, 4096, 4356, 4624, 4900, 5184, 5476, 5776, 6084, 6400, 6724, 7056, 7396, 7744, 8100, 8464, 8836, 9216, 9604, 10000)

scala> var total=0

total: Int = 0
```

scala> for(e <-m) total+=e

scala> total

res45: Int = 171700

scala> val r=m.reduce((total,element)=> total +element)

r: Int = 171700

scala> f.sum

res46: Int = 2550

scala> m.sum

res47: Int = 171700

scala> val s= l.toSet

s: scala.collection.immutable.Set[Int] = Set(69, 88, 5, 10, 56, 42, 24, 37, 25, 52, 14, 20, 46, 93, 57, 78, 29, 84, 61, 89, 1, 74, 6, 60, 85, 28, 38, 70, 21, 33, 92, 65, 97, 9, 53, 77, 96, 13, 41, 73, 2, 32, 34, 45, 64, 17, 22, 44, 59, 27, 71, 12, 54, 49, 86, 81, 76, 7, 39, 98, 91, 66, 3, 80, 35, 48, 63, 18, 95, 50, 67, 16, 31, 11, 72, 43, 99, 87, 40, 26, 55, 23, 8, 75, 58, 82, 36, 30, 51, 19, 4, 79, 94, 47, 15, 68, 62, 90, 83, 100)

scala> val s= l.toSet.

& andThen drop foldLeft inits mkString sameElements subsets toIterator unzip3

&~ apply dropRight foldRight intersect nonEmpty scan sum toList view

+ canEqual dropWhile forall isEmpty par scanLeft tail toMap withFilter

++ collect empty foreach isTraversableAgain partition scanRight tails toSeq zip

++: collectFirst equals genericBuilder iterator product seq take toSet zipAll

- companion exists groupBy last reduce size takeRight toStream zipWithIndex

-- compose filter grouped lastOption reduceLeft slice takeWhile toString |

/: contains filterNot hasDefiniteSize map reduceLeftOption sliding to toTraversable

:\ copyToArray find hashCode max reduceOption span toArray toVector

WithFilter copyToBuffer flatMap head maxBy reduceRight splitAt toBuffer transpose

addString count flatten headOption min reduceRightOption stringPrefix toIndexedSeq union

aggregate diff fold init minBy repr subsetOf toIterable unzip

scala> val s= l.toList.

++ combinations filter indexOf lengthCompare product sameElements tail toTraversable

++: companion filterNot indexOfSlice lift productArity scan tails toVector

+: compose find indexWhere map productElement scanLeft take transpose

/: contains flatMap indices mapConserve productIterator scanRight takeRight union

:+ containsSlice flatten init max productPrefix segmentLength takeWhile unzip

:: copyToArray fold inits maxBy reduce seq to unzip3

::: copyToBuffer foldLeft intersect min reduceLeft size toArray updated

:\ corresponds foldRight isDefinedAt minBy reduceLeftOption slice toBuffer view

WithFilter count forall isEmpty mkString reduceOption sliding toIndexedSeq withFilter

addString diff foreach isTraversableAgain nonEmpty reduceRight sortBy toIterable zip

aggregate distinct genericBuilder iterator orElse reduceRightOption sortWith toIterator zipAll

andThen drop groupBy last padTo repr sorted toList zipWithIndex

apply dropRight grouped lastIndexOf par reverse span toMap

applyOrElse dropWhile hasDefiniteSize lastIndexOfSlice partition reverseIterator splitAt toSeq

canEqual endsWith hashCode lastIndexWhere patch reverseMap startsWith toSet

collect equals head lastOption permutations reverse_::: stringPrefix toStream

collectFirst exists headOption length prefixLength runWith sum toString

scala> val s= l.toList.

!= apply drop foreach isDefinedAt min productIterator scanRight take toVector

applyOrElse dropRight formatted isEmpty minBy productPrefix segmentLength takeRight transpose

+ asInstanceOf dropWhile genericBuilder isInstanceOf mkString reduce seq takeWhile union

++ canEqual endsWith getClass isTraversableAgain ne reduceLeft size to unzip

++: collect ensuring groupBy iterator nonEmpty reduceLeftOption slice toArray unzip3

+: collectFirst eq grouped last notify reduceOption sliding toBuffer updated

-> combinations equals hasDefiniteSize lastIndexOf notifyAll reduceRight sortBy toIndexedSeq view

/: companion exists hashCode lastIndexOfSlice orElse reduceRightOption sortWith toIterable wait

:+ compose filter head lastIndexWhere padTo repr sorted toIterator withFilter

:: contains filterNot headOption lastOption par reverse span toList zip

::: containsSlice find indexOf length partition reverseIterator splitAt toMap zipAll

:\ copyToArray flatMap indexOfSlice lengthCompare patch reverseMap startsWith toParArray zipWithIndex

== copyToBuffer flatten indexWhere lift permutations reverse_::: stringPrefix toSeq →

WithFilter corresponds fold indices map prefixLength runWith sum toSet

addString count foldLeft init mapConserve product sameElements synchronized toStream

aggregate diff foldRight inits max productArity scan tail toString

andThen distinct forall intersect maxBy productElement scanLeft tails toTraversable

scala> val s= l.toList.sorted

s: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)

scala> import scala.io.source

<console>:11: error: object source is not a member of package io

import scala.io.source

^

scala> import scala.io.Source

import scala.io.Source

scala> val orderitems =Source.

DefaultBufSize fromBytes fromChars fromInputStream fromRawBytes fromString fromURL

createBufferedSource fromChar fromFile fromIterable fromResource fromURI stdin

scala> val orderitems =Source.fromFile("/home/training/data-master/retail_db/order_items/part-00000").

++ buffered descr forall map patch reportWarning takeWhile toString

/: bufferedReader drop foreach max pos reset to toTraversable

:\ ch dropWhile getLines maxBy product sameElements toArray toVector

BufferedLineIterator close duplicate grouped min reader scanLeft toBuffer withClose

GroupedIterator codec exists hasDefiniteSize minBy reduce scanRight toIndexedSeq withDescription

LineIterator collect filter hasNext mkString reduceLeft seq toIterable withFilter

NoPositioner collectFirst filterNot indexOf nerrors reduceLeftOption size toIterator withPositioning

Positioner contains find indexWhere next reduceOption slice toList withReset

RelaxedPosition copyToArray flatMap isEmpty nonEmpty reduceRight sliding toMap zip

RelaxedPositioner copyToBuffer fold isTraversableAgain nwarnings reduceRightOption span toSeq zipAll

addString corresponds foldLeft iter padTo report sum toSet zipWithIndex

aggregate count foldRight length partition reportError take toStream

```
scala> val orderitems =Source.fromFile("/home/training/data-master/retail_db/order_items/part-00000").getLines

orderitems: Iterator[String] = non-empty iterator


scala> orderitems.take(10)

res48: Iterator[String] = non-empty iterator


scala> orderitems.take(10).

++ contains filter grouped max patch scanLeft takeWhile toSeq zipWithIndex

/: copyToArray filterNot hasDefiniteSize maxBy product scanRight to toSet

:\ copyToBuffer find hasNext min reduce seq toArray toStream

GroupedIterator corresponds flatMap indexOf minBy reduceLeft size toBuffer toString

addString count fold indexWhere mkString reduceLeftOption slice toIndexedSeq toTraversable

aggregate drop foldLeft isEmpty next reduceOption sliding toIterable toVector

buffered dropWhile foldRight isTraversableAgain nonEmpty reduceRight span toIterator withFilter

collect duplicate forall length padTo reduceRightOption sum toList zip

collectFirst exists foreach map partition sameElements take toMap zipAll


scala> orderitems.take(10).foreach(println)
```

1,1,957,1,299.98,299.98

2,2,1073,1,199.99,199.99

3,2,502,5,250.0,50.0

4,2,403,1,129.99,129.99

5,4,897,2,49.98,24.99

6,4,365,5,299.95,59.99

7,4,502,3,150.0,50.0

8,4,1014,4,199.92,49.98

9,5,957,1,299.98,299.98

10,5,365,5,299.95,59.99

```
scala> val t = (1,1,957,1,299.98,299.98)
t: (Int, Int, Int, Int, Double, Double) = (1,1,957,1,299.98,299.98)

scala> t.
_1 _2 _3 _4 _5 _6 canEqual copy equals hashCode productArity productElement productIterator productPrefix toString

scala> t.t._1
<console>:14: error: value t is not a member of (Int, Int, Int, Int, Double, Double)
       t.t._1
         ^

scala> t._1
res51: Int = 1
```

```
scala> t._2

res52: Int = 1


scala> print(t)

(1,1,957,1,299.98,299.98)

scala> val orderitems =Source.fromFile("/home/training/data-
master/retail_db/order_items/part-00000").getLines.toList

orderitems: List[String] = List(1,1,957,1,299.98,299.98, 2,2,1073,1,199.99,199.99,
3,2,502,5,250.0,50.0, 4,2,403,1,129.99,129.99, 5,4,897,2,49.98,24.99, 6,4,365,5,299.95,59.99,
7,4,502,3,150.0,50.0, 8,4,1014,4,199.92,49.98, 9,5,957,1,299.98,299.98,
10,5,365,5,299.95,59.99, 11,5,1014,2,99.96,49.98, 12,5,957,1,299.98,299.98,
13,5,403,1,129.99,129.99, 14,7,1073,1,199.99,199.99, 15,7,957,1,299.98,299.98,
16,7,926,5,79.95,15.99, 17,8,365,3,179.97,59.99, 18,8,365,5,299.95,59.99,
19,8,1014,4,199.92,49.98, 20,8,502,1,50.0,50.0, 21,9,191,2,199.98,99.99,
22,9,1073,1,199.99,199.99, 23,9,1073,1,199.99,199.99, 24,10,1073,1,199.99,199.99,
25,10,1014,2,99.96,49.98, 26,10,403,1,129.99,129.99, 27,10,917,1,21.99,21.99,
28,10,1073,1,199.99,199.99, 29,11,365,1,59.99,59.99, 30,11,6...


scala> val orderItemsFilter = orderItems.filter

<console>:12: error: not found: value orderItems

      val orderItemsFilter = orderItems.filter

                             ^


scala> val orderItemsFilter = orderItems.filter

<console>:12: error: not found: value orderItems

      val orderItemsFilter = orderItems.filter

                             ^
```

```
scala> val orderitems =Source.fromFile("/home/training/data-master/retail_db/order_items/part-00000").getLines.toList

orderitems: List[String] = List(1,1,957,1,299.98,299.98, 2,2,1073,1,199.99,199.99, 3,2,502,5,250.0,50.0, 4,2,403,1,129.99,129.99, 5,4,897,2,49.98,24.99, 6,4,365,5,299.95,59.99, 7,4,502,3,150.0,50.0, 8,4,1014,4,199.92,49.98, 9,5,957,1,299.98,299.98, 10,5,365,5,299.95,59.99, 11,5,1014,2,99.96,49.98, 12,5,957,1,299.98,299.98, 13,5,403,1,129.99,129.99, 14,7,1073,1,199.99,199.99, 15,7,957,1,299.98,299.98, 16,7,926,5,79.95,15.99, 17,8,365,3,179.97,59.99, 18,8,365,5,299.95,59.99, 19,8,1014,4,199.92,49.98, 20,8,502,1,50.0,50.0, 21,9,191,2,199.98,99.99, 22,9,1073,1,199.99,199.99, 23,9,1073,1,199.99,199.99, 24,10,1073,1,199.99,199.99, 25,10,1014,2,99.96,49.98, 26,10,403,1,129.99,129.99, 27,10,917,1,21.99,21.99, 28,10,1073,1,199.99,199.99, 29,11,365,1,59.99,59.99, 30,11,6...


scala> val orderItemsFilter = orderItems.

     | .

<console>:2: error: identifier expected but '.' found.

       .

       ^


scala> val orderItemsFilter = orderitems.

++          compose       flatten        intersect         mkString         reduceRightOption   span
toSet

++:         contains      fold           isDefinedAt       nonEmpty         repr                splitAt
toStream

+:          containsSlice   foldLeft     isEmpty           orElse           reverse             startsWith
toString

/:          copyToArray   foldRight      isTraversableAgain   padTo         reverseIterator
stringPrefix   toTraversable
```

:+ copyToBuffer forall iterator par reverseMap sum toVector

:: corresponds foreach last partition reverse_::: tail transpose

::: count genericBuilder lastIndexOf patch runWith tails union

:\ diff groupBy lastIndexOfSlice permutations sameElements take unzip

WithFilter distinct grouped lastIndexWhere prefixLength scan takeRight unzip3

addString drop hasDefiniteSize lastOption product scanLeft takeWhile updated

aggregate dropRight hashCode length productArity scanRight to view

andThen dropWhile head lengthCompare productElement segmentLength toArray withFilter

apply endsWith headOption lift productIterator seq toBuffer zip

applyOrElse equals indexOf map productPrefix size toIndexedSeq zipAll

canEqual exists indexOfSlice mapConserve reduce slice toIterable zipWithIndex

collect filter indexWhere max reduceLeft sliding toIterator

collectFirst filterNot indices maxBy reduceLeftOption sortBy toList

combinations find init min reduceOption sortWith toMap

companion flatMap inits minBy reduceRight sorted toSeq

scala> val orderItemsFilter = orderitems.filter

```
<console>:13: error: missing argument list for method filter in trait TraversableLike
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `filter _` or `filter(_)` instead of `filter`.
       val orderItemsFilter = orderitems.filter
                                         ^

scala> val orderItemsFilter = orderitems.filter
<console>:13: error: missing argument list for method filter in trait TraversableLike
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `filter _` or `filter(_)` instead of `filter`.
       val orderItemsFilter = orderitems.filter
                                         ^

scala> val orderItemsFilter = orderitems.filter
<console>:13: error: missing argument list for method filter in trait TraversableLike
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `filter _` or `filter(_)` instead of `filter`.
       val orderItemsFilter = orderitems.filter
                                         ^

scala> val orderItemsFilter = orderitems.filter
<console>:13: error: missing argument list for method filter in trait TraversableLike
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `filter _` or `filter(_)` instead of `filter`.
```

```
    val orderItemsFilter = orderitems.filter
                           ^

scala> val orderItemsFilter = orderitems.filter
<console>:13: error: missing argument list for method filter in trait TraversableLike
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `filter _` or `filter(_)` instead of `filter`.
    val orderItemsFilter = orderitems.filter
                           ^

scala> val orderItemsFilter = orderitems.filter
filter   filterNot

scala> val orderItemsFilter = orderitems.filter
<console>:13: error: missing argument list for method filter in trait TraversableLike
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `filter _` or `filter(_)` instead of `filter`.
    val orderItemsFilter = orderitems.filter
                           ^

scala> val orderItemsFilter = orderitems.filter()
<console>:13: error: not enough arguments for method filter: (p: String => Boolean)List[String].
Unspecified value parameter p.
    val orderItemsFilter = orderitems.filter()
```

^

scala> val orderItemsFilter = orderitems.filter(orderItem => orderItem.split(",")(1).toInt == 2)

orderItemsFilter: List[String] = List(2,2,1073,1,199.99,199.99, 3,2,502,5,250.0,50.0, 4,2,403,1,129.99,129.99)

scala> val orderItemsMap = orderItemsFilter.map(orderItem => orderItem.split(",")(4).toFloat)

orderItemsMap: List[Float] = List(199.99, 250.0, 129.99)

scala> orderItemsMap.sum

res54: Float = 579.98

scala> orderItemsMap.reduce

<console>:14: error: missing argument list for method reduce in trait TraversableOnce

Unapplied methods are only converted to functions when a function type is expected.

You can make this conversion explicit by writing `reduce _` or `reduce(_)` instead of `reduce`.

orderItemsMap.reduce

^

scala> orderItemsMap.reduce

reduce reduceLeft reduceLeftOption reduceOption reduceRight reduceRightOption

scala> orderItemsMap.

++ compose flatten intersect mkString reduceRightOption span toSet

++: contains fold isDefinedAt nonEmpty repr splitAt toStream

+: containsSlice foldLeft isEmpty orElse reverse startsWith toString

/: copyToArray foldRight isTraversableAgain padTo reverseIterator stringPrefix toTraversable

:+ copyToBuffer forall iterator par reverseMap sum toVector

:: corresponds foreach last partition reverse_::: tail transpose

::: count genericBuilder lastIndexOf patch runWith tails union

:\ diff groupBy lastIndexOfSlice permutations sameElements take unzip

WithFilter distinct grouped lastIndexWhere prefixLength scan takeRight unzip3

addString drop hasDefiniteSize lastOption product scanLeft takeWhile updated

aggregate dropRight hashCode length productArity scanRight to view

andThen dropWhile head lengthCompare productElement segmentLength toArray withFilter

apply endsWith headOption lift productIterator seq toBuffer zip

applyOrElse equals indexOf map productPrefix size toIndexedSeq zipAll

canEqual exists indexOfSlice mapConserve reduce slice toIterable zipWithIndex

collect filter indexWhere max reduceLeft sliding toIterator

```
collectFirst  filterNot  indices  maxBy  reduceLeftOption  sortBy  toList
combinations  find  init  min  reduceOption  sortWith  toMap
companion  flatMap  inits  minBy  reduceRight  sorted  toSeq

scala> orderItemsMap.reduce
reduce  reduceLeft  reduceLeftOption  reduceOption  reduceRight  reduceRightOption

scala> orderItemsMap.reduce
<console>:14: error: missing argument list for method reduce in trait TraversableOnce
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `reduce _` or `reduce(_)` instead of `reduce`.
       orderItemsMap.reduce
                     ^

scala> orderItemsMap.reduce((total,orderItemSubtotal) => total + orderItem)
<console>:15: error: not found: value orderItem
       orderItemsMap.reduce((total,orderItemSubtotal) => total + orderItem)
                                                          ^

scala> orderItemsMap.reduce((total,orderItemSubtotal) => total + orderItemSubtotal)
res58: Float = 579.98

scala> val orderItemsFilter = orderitems.filter(_.split(",")(1).toInt == 2)
orderItemsFilter: List[String] = List(2,2,1073,1,199.99,199.99, 3,2,502,5,250.0,50.0,
4,2,403,1,129.99,129.99)
```

```
scala> orderItemsMap.reduce(_+_)

res59: Float = 579.98
```

Data Frames:

Read json file:

68390,2014-04-26 00:00:00.0,8939,PENDING_PAYMENT

68391,2014-04-27 00:00:00.0,4852,PENDING_PAYMENT

68392,2014-04-27 00:00:00.0,8539,SUSPECTED_FRAUD

68393,2014-04-27 00:00:00.0,2928,PENDING_PAYMENT

68394,2014-04-27 00:00:00.0,7939,PENDING_PAYMENT

68395,2014-04-28 00:00:00.0,9263,COMPLETE

68396,2014-04-28 00:00:00.0,11163,COMPLETE

68397,2014-04-28 00:00:00.0,10438,PENDING

68398,2014-04-28 00:00:00.0,2789,COMPLETE

68399,2014-04-28 00:00:00.0,1034,CLOSED

68400,2014-04-28 00:00:00.0,8311,SUSPECTED_FRAUD

68401,2014-04-29 00:00:00.0,5433,PROCESSING

68402,2014-04-30 00:00:00.0,11123,CLOSED

68403,2014-04-30 00:00:00.0,9957,COMPLETE

68404,2014-04-30 00:00:00.0,10728,COMPLETE

68405,2014-04-30 00:00:00.0,6224,PENDING_PAYMENT

68406,2014-04-30 00:00:00.0,8833,COMPLETE

68407,2014-04-30 00:00:00.0,11418,COMPLETE

68408,2014-05-01 00:00:00.0,934,COMPLETE

68409,2014-05-01 00:00:00.0,11684,PENDING_PAYMENT

68410,2014-05-01 00:00:00.0,11991,ON_HOLD

68411,2014-05-02 00:00:00.0,1472,COMPLETE

68412,2014-05-03 00:00:00.0,9669,SUSPECTED_FRAUD

68413,2014-05-03 00:00:00.0,11733,COMPLETE

68414,2014-05-03 00:00:00.0,10967,COMPLETE

68415,2014-05-03 00:00:00.0,1078,COMPLETE

68416,2014-05-04 00:00:00.0,7102,SUSPECTED_FRAUD

68417,2014-05-04 00:00:00.0,7553,PENDING_PAYMENT

68418,2014-05-04 00:00:00.0,2804,PAYMENT_REVIEW

68419,2014-05-05 00:00:00.0,8718,COMPLETE

68420,2014-05-05 00:00:00.0,1550,CLOSED

68421,2014-05-05 00:00:00.0,10619,PENDING

68422,2014-05-05 00:00:00.0,3870,CANCELED

68423,2014-05-06 00:00:00.0,9004,PENDING

68424,2014-05-07 00:00:00.0,4320,CLOSED

68425,2014-05-07 00:00:00.0,10080,PENDING_PAYMENT

68426,2014-05-07 00:00:00.0,11361,CANCELED

68427,2014-05-07 00:00:00.0,2484,PENDING_PAYMENT

68428,2014-05-08 00:00:00.0,3471,PROCESSING

68429,2014-05-09 00:00:00.0,11170,COMPLETE

68430,2014-05-09 00:00:00.0,7085,COMPLETE

68431,2014-05-09 00:00:00.0,9688,PROCESSING

68432,2014-05-09 00:00:00.0,2339,COMPLETE

68433,2014-05-10 00:00:00.0,4557,COMPLETE

68434,2014-05-11 00:00:00.0,6812,PROCESSING

68435,2014-05-11 00:00:00.0,10441,CLOSED

68436,2014-05-11 00:00:00.0,6589,PENDING_PAYMENT

68437,2014-05-11 00:00:00.0,3413,CANCELED

68438,2014-05-11 00:00:00.0,5710,PROCESSING

68439,2014-05-11 00:00:00.0,10490,COMPLETE

68440,2014-05-12 00:00:00.0,791,PENDING_PAYMENT

68441,2014-05-12 00:00:00.0,6829,COMPLETE

68442,2014-05-12 00:00:00.0,11724,PROCESSING

68443,2014-05-12 00:00:00.0,8259,PROCESSING

68444,2014-05-12 00:00:00.0,3399,CANCELED

68445,2014-05-12 00:00:00.0,12064,PENDING

68446,2014-05-13 00:00:00.0,4147,PENDING_PAYMENT

68447,2014-05-14 00:00:00.0,2626,COMPLETE

68448,2014-05-14 00:00:00.0,2287,CLOSED

68449,2014-05-14 00:00:00.0,9844,PROCESSING

68450,2014-05-14 00:00:00.0,10194,COMPLETE

68451,2014-05-14 00:00:00.0,8929,CLOSED

68452,2014-05-15 00:00:00.0,636,COMPLETE

68453,2014-05-15 00:00:00.0,4400,PENDING_PAYMENT

68454,2014-05-15 00:00:00.0,9297,CANCELED

68455,2014-05-15 00:00:00.0,921,PROCESSING

68456,2014-05-15 00:00:00.0,8208,COMPLETE

68457,2014-05-16 00:00:00.0,2562,COMPLETE

68458,2014-05-17 00:00:00.0,10089,PENDING_PAYMENT

68459,2014-05-17 00:00:00.0,10423,COMPLETE

68460,2014-05-17 00:00:00.0,10096,CLOSED

68461,2014-05-17 00:00:00.0,759,ON_HOLD

68462,2014-05-17 00:00:00.0,10438,PENDING_PAYMENT

68463,2014-05-17 00:00:00.0,11028,PROCESSING

68464,2014-05-18 00:00:00.0,11001,PENDING_PAYMENT

68465,2014-05-18 00:00:00.0,10745,PENDING_PAYMENT

68466,2014-05-18 00:00:00.0,334,PENDING_PAYMENT

68467,2014-05-18 00:00:00.0,5214,CANCELED

68468,2014-05-18 00:00:00.0,5213,COMPLETE

68469,2014-05-18 00:00:00.0,10964,CLOSED

68470,2014-05-19 00:00:00.0,3270,PENDING_PAYMENT

68471,2014-05-19 00:00:00.0,7816,CLOSED

68472,2014-05-19 00:00:00.0,2741,COMPLETE

68473,2014-05-19 00:00:00.0,10584,COMPLETE

68474,2014-05-19 00:00:00.0,5038,COMPLETE

68475,2014-05-20 00:00:00.0,883,PENDING_PAYMENT

68476,2014-05-20 00:00:00.0,9846,PROCESSING

68477,2014-05-20 00:00:00.0,2076,CLOSED

68478,2014-05-21 00:00:00.0,11716,ON_HOLD

68479,2014-05-21 00:00:00.0,260,PENDING

68480,2014-05-21 00:00:00.0,8859,COMPLETE

68481,2014-05-21 00:00:00.0,2000,PENDING_PAYMENT

68482,2014-05-22 00:00:00.0,9141,COMPLETE

68483,2014-05-22 00:00:00.0,9974,ON_HOLD

68484,2014-05-22 00:00:00.0,2167,PROCESSING

68485,2014-05-23 00:00:00.0,6496,PROCESSING

68486,2014-05-23 00:00:00.0,8791,PROCESSING

68487,2014-05-23 00:00:00.0,11710,CLOSED

68488,2014-05-23 00:00:00.0,2988,COMPLETE

68489,2014-05-23 00:00:00.0,2180,CLOSED

68490,2014-05-24 00:00:00.0,7369,PENDING

68491,2014-05-24 00:00:00.0,7736,PENDING_PAYMENT

68492,2014-05-24 00:00:00.0,4331,COMPLETE

68493,2014-05-24 00:00:00.0,7013,CLOSED

68494,2014-05-24 00:00:00.0,10053,PENDING

68495,2014-05-24 00:00:00.0,11792,COMPLETE

68496,2014-05-25 00:00:00.0,3156,PENDING_PAYMENT

68497,2014-05-25 00:00:00.0,1973,PENDING_PAYMENT

68498,2014-05-26 00:00:00.0,11257,PENDING_PAYMENT

68499,2014-05-26 00:00:00.0,10220,PENDING_PAYMENT

68500,2014-05-27 00:00:00.0,3459,COMPLETE

68501,2014-05-28 00:00:00.0,2394,COMPLETE

68502,2014-05-29 00:00:00.0,9730,PROCESSING

68503,2014-05-29 00:00:00.0,9830,CLOSED

68504,2014-05-29 00:00:00.0,9094,CLOSED

68505,2014-05-29 00:00:00.0,378,COMPLETE

68506,2014-05-29 00:00:00.0,10527,SUSPECTED_FRAUD

68507,2014-05-30 00:00:00.0,8,PENDING

68508,2014-05-30 00:00:00.0,3081,PENDING_PAYMENT

68509,2014-05-31 00:00:00.0,628,COMPLETE

68510,2014-05-31 00:00:00.0,11175,COMPLETE

68511,2014-05-31 00:00:00.0,5012,PENDING_PAYMENT

68512,2014-06-01 00:00:00.0,1137,PENDING_PAYMENT

68513,2014-06-01 00:00:00.0,9897,COMPLETE

68514,2014-06-01 00:00:00.0,4685,COMPLETE

68515,2014-06-02 00:00:00.0,2105,PROCESSING

68516,2014-06-03 00:00:00.0,5372,CANCELED

68517,2014-06-04 00:00:00.0,931,PENDING_PAYMENT

68518,2014-06-04 00:00:00.0,6917,PROCESSING

68519,2014-06-04 00:00:00.0,9047,COMPLETE

68520,2014-06-04 00:00:00.0,12309,ON_HOLD

68521,2014-06-05 00:00:00.0,11872,PROCESSING

68522,2014-06-05 00:00:00.0,880,SUSPECTED_FRAUD

68523,2014-06-05 00:00:00.0,1432,PENDING_PAYMENT

68524,2014-06-05 00:00:00.0,2202,COMPLETE

68525,2014-06-05 00:00:00.0,6353,COMPLETE

68526,2014-06-05 00:00:00.0,3387,PENDING

68527,2014-06-06 00:00:00.0,5337,CLOSED

68528,2014-06-06 00:00:00.0,10183,CANCELED

68529,2014-06-06 00:00:00.0,11309,COMPLETE

68530,2014-06-06 00:00:00.0,7019,COMPLETE

68531,2014-06-06 00:00:00.0,2120,COMPLETE

68532,2014-06-06 00:00:00.0,77,CLOSED

68533,2014-06-07 00:00:00.0,10699,PENDING_PAYMENT

68534,2014-06-08 00:00:00.0,5168,SUSPECTED_FRAUD

68535,2014-06-08 00:00:00.0,8652,COMPLETE

68536,2014-06-08 00:00:00.0,1231,COMPLETE

68537,2014-06-08 00:00:00.0,2596,PENDING

68538,2014-06-09 00:00:00.0,915,PENDING_PAYMENT

68539,2014-06-10 00:00:00.0,7445,PENDING_PAYMENT

68540,2014-06-10 00:00:00.0,5211,PENDING_PAYMENT

68541,2014-06-11 00:00:00.0,7554,PENDING

68542,2014-06-11 00:00:00.0,5035,ON_HOLD

68543,2014-06-11 00:00:00.0,8952,COMPLETE

68544,2014-06-12 00:00:00.0,10605,PROCESSING

68545,2014-06-12 00:00:00.0,3582,PENDING_PAYMENT

68546,2014-06-12 00:00:00.0,6405,PENDING_PAYMENT

68547,2014-06-12 00:00:00.0,8457,PENDING

68548,2014-06-12 00:00:00.0,9245,COMPLETE

68549,2014-06-12 00:00:00.0,3733,COMPLETE

68550,2014-06-13 00:00:00.0,11785,PENDING_PAYMENT

68551,2014-06-13 00:00:00.0,3727,PENDING

68552,2014-06-13 00:00:00.0,1763,COMPLETE

68553,2014-06-13 00:00:00.0,10037,PENDING_PAYMENT

68554,2014-06-13 00:00:00.0,1894,PROCESSING

68555,2014-06-14 00:00:00.0,4454,SUSPECTED_FRAUD

68556,2014-06-14 00:00:00.0,618,COMPLETE

68557,2014-06-14 00:00:00.0,10109,CANCELED

68558,2014-06-14 00:00:00.0,6881,COMPLETE

68559,2014-06-14 00:00:00.0,12275,PENDING

68560,2014-06-15 00:00:00.0,2572,PENDING_PAYMENT

68561,2014-06-15 00:00:00.0,6626,COMPLETE

68562,2014-06-15 00:00:00.0,6385,PAYMENT_REVIEW

68563,2014-06-16 00:00:00.0,4378,PENDING_PAYMENT

68564,2014-06-17 00:00:00.0,5272,COMPLETE

68565,2014-06-17 00:00:00.0,4715,PENDING

68566,2014-06-17 00:00:00.0,2214,CLOSED

68567,2014-06-18 00:00:00.0,3951,COMPLETE

68568,2014-06-19 00:00:00.0,7418,PAYMENT_REVIEW

68569,2014-06-19 00:00:00.0,2755,COMPLETE

68570,2014-06-19 00:00:00.0,6015,CLOSED

68571,2014-06-19 00:00:00.0,2796,CANCELED

68572,2014-06-19 00:00:00.0,2244,COMPLETE

68573,2014-06-20 00:00:00.0,2219,COMPLETE

68574,2014-06-20 00:00:00.0,1837,PENDING

68575,2014-06-20 00:00:00.0,3873,COMPLETE

68576,2014-06-20 00:00:00.0,6184,PROCESSING

68577,2014-06-21 00:00:00.0,11615,PENDING

68578,2014-06-21 00:00:00.0,4350,COMPLETE

68579,2014-06-21 00:00:00.0,9038,PENDING

68580,2014-06-22 00:00:00.0,4751,PENDING_PAYMENT

68581,2014-06-22 00:00:00.0,6020,PROCESSING

68582,2014-06-22 00:00:00.0,10705,CLOSED

68583,2014-06-22 00:00:00.0,11149,PENDING

68584,2014-06-22 00:00:00.0,6129,PROCESSING

68585,2014-06-23 00:00:00.0,6999,COMPLETE

68586,2014-06-23 00:00:00.0,2448,PENDING

68587,2014-06-23 00:00:00.0,5976,PENDING

68588,2014-06-23 00:00:00.0,11985,COMPLETE

68589,2014-06-24 00:00:00.0,11503,COMPLETE

68590,2014-06-24 00:00:00.0,9302,PENDING

68591,2014-06-24 00:00:00.0,8704,PENDING_PAYMENT

68592,2014-06-24 00:00:00.0,8456,COMPLETE

68593,2014-06-24 00:00:00.0,7043,ON_HOLD

68594,2014-06-25 00:00:00.0,7802,COMPLETE

68595,2014-06-25 00:00:00.0,12055,COMPLETE

68596,2014-06-25 00:00:00.0,4246,ON_HOLD

68597,2014-06-25 00:00:00.0,9506,PENDING_PAYMENT

68598,2014-06-26 00:00:00.0,10947,COMPLETE

68599,2014-06-26 00:00:00.0,10433,COMPLETE

68600,2014-06-27 00:00:00.0,4206,PROCESSING

68601,2014-06-27 00:00:00.0,687,PROCESSING

68602,2014-06-27 00:00:00.0,10412,PENDING

68603,2014-06-27 00:00:00.0,5259,COMPLETE

68604,2014-06-27 00:00:00.0,8652,COMPLETE

68605,2014-06-28 00:00:00.0,11454,COMPLETE

68606,2014-06-28 00:00:00.0,2253,SUSPECTED_FRAUD

68607,2014-06-28 00:00:00.0,1947,COMPLETE

68608,2014-06-28 00:00:00.0,419,PENDING_PAYMENT

68609,2014-06-28 00:00:00.0,3342,PENDING_PAYMENT

68610,2014-06-28 00:00:00.0,5200,ON_HOLD

68611,2014-06-29 00:00:00.0,11730,CLOSED

68612,2014-06-29 00:00:00.0,1614,COMPLETE

68613,2014-06-29 00:00:00.0,2844,PROCESSING

68614,2014-06-29 00:00:00.0,272,COMPLETE

68615,2014-06-29 00:00:00.0,5250,CLOSED

68616,2014-06-29 00:00:00.0,1475,PENDING_PAYMENT

68617,2014-06-30 00:00:00.0,6188,COMPLETE

68618,2014-07-01 00:00:00.0,11936,PENDING

68619,2014-07-01 00:00:00.0,2056,PROCESSING

68620,2014-07-01 00:00:00.0,11490,PROCESSING

68621,2014-07-02 00:00:00.0,4266,ON_HOLD

68622,2014-07-02 00:00:00.0,8186,CLOSED

68623,2014-07-02 00:00:00.0,6574,COMPLETE

68624,2014-07-02 00:00:00.0,8379,PROCESSING

68625,2014-07-02 00:00:00.0,1765,PROCESSING

68626,2014-07-02 00:00:00.0,5548,CLOSED

68627,2014-07-03 00:00:00.0,5493,CLOSED

68628,2014-07-03 00:00:00.0,223,PENDING

68629,2014-07-04 00:00:00.0,7369,PENDING

68630,2014-07-04 00:00:00.0,8382,CLOSED

68631,2014-07-04 00:00:00.0,697,PENDING

68632,2014-07-04 00:00:00.0,10713,CLOSED

68633,2014-07-05 00:00:00.0,10481,PENDING_PAYMENT

68634,2014-07-05 00:00:00.0,4730,COMPLETE

68635,2014-07-05 00:00:00.0,9260,CLOSED

68636,2014-07-06 00:00:00.0,8537,PROCESSING

68637,2014-07-07 00:00:00.0,7447,PROCESSING

68638,2014-07-07 00:00:00.0,11631,COMPLETE

68639,2014-07-07 00:00:00.0,5091,COMPLETE

68640,2014-07-07 00:00:00.0,11447,PENDING_PAYMENT

68641,2014-07-07 00:00:00.0,221,PENDING_PAYMENT

68642,2014-07-08 00:00:00.0,6049,CLOSED

68643,2014-07-08 00:00:00.0,3079,COMPLETE

68644,2014-07-09 00:00:00.0,4115,PENDING_PAYMENT

68645,2014-07-09 00:00:00.0,10013,PENDING_PAYMENT

68646,2014-07-09 00:00:00.0,7631,PENDING

68647,2014-07-09 00:00:00.0,5518,COMPLETE

68648,2014-07-09 00:00:00.0,8589,COMPLETE

68649,2014-07-10 00:00:00.0,7664,CLOSED

68650,2014-07-10 00:00:00.0,11193,PENDING

68651,2014-07-10 00:00:00.0,4736,PENDING

68652,2014-07-10 00:00:00.0,2163,COMPLETE

68653,2014-07-11 00:00:00.0,3263,PENDING

68654,2014-07-12 00:00:00.0,9486,PENDING_PAYMENT

68655,2014-07-13 00:00:00.0,84,PENDING_PAYMENT

68656,2014-07-13 00:00:00.0,12154,PENDING_PAYMENT

68657,2014-07-13 00:00:00.0,9211,CLOSED

68658,2014-07-14 00:00:00.0,7177,PENDING_PAYMENT

68659,2014-07-14 00:00:00.0,11092,COMPLETE

68660,2014-07-15 00:00:00.0,8420,PENDING_PAYMENT

68661,2014-07-15 00:00:00.0,10585,COMPLETE

68662,2014-07-15 00:00:00.0,8643,CLOSED

68663,2014-07-15 00:00:00.0,6176,PENDING_PAYMENT

68664,2014-07-15 00:00:00.0,12430,PROCESSING

68665,2014-07-16 00:00:00.0,368,COMPLETE

68666,2014-07-17 00:00:00.0,3430,PENDING

68667,2014-07-17 00:00:00.0,2920,CLOSED

68668,2014-07-17 00:00:00.0,6140,CLOSED

68669,2014-07-17 00:00:00.0,6668,PROCESSING

68670,2014-07-17 00:00:00.0,10673,ON_HOLD

68671,2014-07-17 00:00:00.0,3046,PENDING

68672,2014-07-18 00:00:00.0,5908,PROCESSING

68673,2014-07-18 00:00:00.0,3372,PENDING

68674,2014-07-18 00:00:00.0,5375,COMPLETE

68675,2014-07-19 00:00:00.0,1617,CLOSED

68676,2014-07-19 00:00:00.0,576,PENDING

68677,2014-07-20 00:00:00.0,5001,COMPLETE

68678,2014-07-20 00:00:00.0,684,PENDING_PAYMENT

68679,2014-07-20 00:00:00.0,9001,ON_HOLD

68680,2014-07-20 00:00:00.0,3636,PENDING_PAYMENT

68681,2014-07-20 00:00:00.0,11014,PENDING_PAYMENT

68682,2014-07-21 00:00:00.0,6017,COMPLETE

68683,2014-07-21 00:00:00.0,1338,PENDING_PAYMENT

68684,2014-07-22 00:00:00.0,6787,PENDING

68685,2014-07-23 00:00:00.0,11426,PENDING_PAYMENT

68686,2014-07-23 00:00:00.0,2591,SUSPECTED_FRAUD

68687,2014-07-23 00:00:00.0,2609,CLOSED

68688,2014-07-24 00:00:00.0,5296,COMPLETE

68689,2014-07-24 00:00:00.0,4380,PROCESSING

68690,2014-07-24 00:00:00.0,4303,PROCESSING

68691,2013-07-25 00:00:00.0,9127,CLOSED

68692,2013-07-26 00:00:00.0,11868,COMPLETE

68693,2013-07-27 00:00:00.0,228,PENDING_PAYMENT

68694,2013-07-29 00:00:00.0,2441,COMPLETE

68695,2013-07-30 00:00:00.0,3918,PROCESSING

68696,2013-07-31 00:00:00.0,5155,COMPLETE

68697,2013-08-03 00:00:00.0,9141,COMPLETE

68698,2013-08-07 00:00:00.0,9100,PENDING_PAYMENT

68699,2013-08-08 00:00:00.0,2983,PENDING_PAYMENT

68700,2013-08-11 00:00:00.0,11078,PROCESSING

68701,2013-08-12 00:00:00.0,6209,PROCESSING

68702,2013-08-14 00:00:00.0,2277,PENDING_PAYMENT

68703,2013-08-16 00:00:00.0,9515,COMPLETE

68704,2013-08-18 00:00:00.0,5204,PROCESSING

68705,2013-08-19 00:00:00.0,8681,CLOSED

68706,2013-08-20 00:00:00.0,130,COMPLETE

68707,2013-08-23 00:00:00.0,11730,COMPLETE

68708,2013-08-26 00:00:00.0,8852,ON_HOLD

68709,2013-08-30 00:00:00.0,4756,COMPLETE

68710,2013-08-31 00:00:00.0,9685,COMPLETE

68711,2013-09-01 00:00:00.0,543,COMPLETE

68712,2013-09-03 00:00:00.0,12401,COMPLETE

68713,2013-09-04 00:00:00.0,2361,COMPLETE

68714,2013-09-06 00:00:00.0,8889,PENDING_PAYMENT

68715,2013-09-09 00:00:00.0,12034,PENDING_PAYMENT

68716,2013-09-12 00:00:00.0,8872,PENDING_PAYMENT

68717,2013-09-13 00:00:00.0,1785,CLOSED

68718,2013-09-14 00:00:00.0,6710,SUSPECTED_FRAUD

68719,2013-09-16 00:00:00.0,11204,COMPLETE

68720,2013-09-18 00:00:00.0,5079,CLOSED

68721,2013-09-19 00:00:00.0,8410,COMPLETE

68722,2013-09-20 00:00:00.0,4258,ON_HOLD

68723,2013-09-22 00:00:00.0,11969,PENDING

68724,2013-09-26 00:00:00.0,1148,COMPLETE

68725,2013-10-01 00:00:00.0,7795,PROCESSING

68726,2013-10-02 00:00:00.0,8817,PENDING_PAYMENT

68727,2013-10-05 00:00:00.0,5880,PENDING_PAYMENT

68728,2013-10-07 00:00:00.0,7267,COMPLETE

68729,2013-10-09 00:00:00.0,8043,PENDING_PAYMENT

68730,2013-10-11 00:00:00.0,3568,PENDING_PAYMENT

68731,2013-10-13 00:00:00.0,8102,PENDING_PAYMENT

68732,2013-10-14 00:00:00.0,9990,COMPLETE

68733,2013-10-16 00:00:00.0,12429,CLOSED

68734,2013-10-18 00:00:00.0,8510,ON_HOLD

68735,2013-10-21 00:00:00.0,788,COMPLETE

68736,2013-10-23 00:00:00.0,8462,COMPLETE

68737,2013-10-26 00:00:00.0,10302,PROCESSING

68738,2013-10-27 00:00:00.0,1100,COMPLETE

68739,2013-10-28 00:00:00.0,2528,PENDING

68740,2013-10-29 00:00:00.0,10691,ON_HOLD

68741,2013-10-30 00:00:00.0,5974,PENDING_PAYMENT

68742,2013-10-31 00:00:00.0,197,COMPLETE

68743,2013-11-01 00:00:00.0,2751,CANCELED

68744,2013-11-03 00:00:00.0,10540,PROCESSING

68745,2013-11-04 00:00:00.0,2777,CLOSED

68746,2013-11-05 00:00:00.0,11243,PROCESSING

68747,2013-11-07 00:00:00.0,10192,CLOSED

68748,2013-11-08 00:00:00.0,11421,PENDING_PAYMENT

68749,2013-11-09 00:00:00.0,10868,PENDING_PAYMENT

68750,2013-11-11 00:00:00.0,6336,PENDING

68751,2013-11-12 00:00:00.0,6110,PENDING

68752,2013-11-14 00:00:00.0,8271,PENDING_PAYMENT

68753,2013-11-15 00:00:00.0,6927,COMPLETE

68754,2013-11-16 00:00:00.0,2424,CANCELED

68755,2013-11-17 00:00:00.0,2644,COMPLETE

68756,2013-11-19 00:00:00.0,525,CLOSED

68757,2013-11-20 00:00:00.0,6462,PENDING_PAYMENT

68758,2013-11-23 00:00:00.0,8572,CLOSED

68759,2013-11-27 00:00:00.0,16,COMPLETE

68760,2013-11-30 00:00:00.0,3603,COMPLETE

68761,2013-12-04 00:00:00.0,6402,COMPLETE

68762,2013-12-05 00:00:00.0,1761,PENDING_PAYMENT

68763,2013-12-06 00:00:00.0,241,PENDING

68764,2013-12-08 00:00:00.0,1735,COMPLETE

68765,2013-12-09 00:00:00.0,5243,PROCESSING

68766,2013-12-10 00:00:00.0,382,PROCESSING

68767,2013-12-11 00:00:00.0,6671,CANCELED

68768,2013-12-12 00:00:00.0,4150,COMPLETE

68769,2013-12-13 00:00:00.0,1296,PROCESSING

68770,2013-12-15 00:00:00.0,1087,PENDING_PAYMENT

68771,2013-12-18 00:00:00.0,283,PROCESSING

68772,2013-12-21 00:00:00.0,6054,COMPLETE

68773,2013-12-23 00:00:00.0,10092,CLOSED

68774,2013-12-24 00:00:00.0,4966,COMPLETE

68775,2013-12-26 00:00:00.0,4077,PROCESSING

68776,2013-12-29 00:00:00.0,1638,CLOSED

68777,2014-01-02 00:00:00.0,11022,PENDING_PAYMENT

68778,2014-01-03 00:00:00.0,986,COMPLETE

68779,2014-01-05 00:00:00.0,7172,PROCESSING

68780,2014-01-06 00:00:00.0,4089,PROCESSING

68781,2014-01-08 00:00:00.0,9037,CLOSED

68782,2014-01-10 00:00:00.0,8509,SUSPECTED_FRAUD

68783,2014-01-12 00:00:00.0,9462,PROCESSING

68784,2014-01-14 00:00:00.0,10349,COMPLETE

68785,2014-01-16 00:00:00.0,10778,PENDING

68786,2014-01-19 00:00:00.0,1847,COMPLETE

68787,2014-01-20 00:00:00.0,3219,PROCESSING

68788,2014-01-23 00:00:00.0,11334,COMPLETE

68789,2014-01-25 00:00:00.0,7658,COMPLETE

68790,2014-01-26 00:00:00.0,10302,CLOSED

68791,2014-01-27 00:00:00.0,6524,COMPLETE

68792,2014-01-28 00:00:00.0,9809,CANCELED

68793,2014-01-30 00:00:00.0,5654,COMPLETE

68794,2014-01-31 00:00:00.0,6873,COMPLETE

68795,2014-02-09 00:00:00.0,6950,PENDING_PAYMENT

68796,2014-02-10 00:00:00.0,12377,COMPLETE

68797,2014-02-12 00:00:00.0,5932,PENDING

68798,2014-02-13 00:00:00.0,2595,COMPLETE

68799,2014-02-14 00:00:00.0,11190,COMPLETE

68800,2014-02-17 00:00:00.0,10037,PROCESSING

68801,2014-02-18 00:00:00.0,2079,PENDING_PAYMENT

68802,2014-02-19 00:00:00.0,10670,COMPLETE

68803,2014-02-23 00:00:00.0,9397,CLOSED

68804,2014-02-24 00:00:00.0,1733,PROCESSING

68805,2014-03-01 00:00:00.0,6599,CLOSED

68806,2014-03-02 00:00:00.0,10351,COMPLETE

68807,2014-03-03 00:00:00.0,3509,PROCESSING

68808,2014-03-10 00:00:00.0,2708,PENDING_PAYMENT

68809,2014-03-12 00:00:00.0,5946,ON_HOLD

68810,2014-03-15 00:00:00.0,9394,PROCESSING

68811,2014-03-17 00:00:00.0,3301,COMPLETE

68812,2014-03-18 00:00:00.0,1985,PROCESSING

68813,2014-03-19 00:00:00.0,3988,COMPLETE

68814,2014-03-23 00:00:00.0,11408,PENDING

68815,2014-03-25 00:00:00.0,1078,CLOSED

68816,2014-03-26 00:00:00.0,8769,CANCELED

68817,2014-03-27 00:00:00.0,6704,COMPLETE

68818,2014-03-31 00:00:00.0,12393,PROCESSING

68819,2014-04-03 00:00:00.0,1212,COMPLETE

68820,2014-04-04 00:00:00.0,6358,COMPLETE

68821,2014-04-05 00:00:00.0,2564,COMPLETE

68822,2014-04-06 00:00:00.0,4844,COMPLETE

68823,2014-04-07 00:00:00.0,7864,COMPLETE

68824,2014-04-08 00:00:00.0,876,PENDING_PAYMENT

68825,2014-04-10 00:00:00.0,10201,PENDING_PAYMENT

68826,2014-04-14 00:00:00.0,7958,PROCESSING

68827,2014-04-16 00:00:00.0,8814,CLOSED

68828,2014-04-20 00:00:00.0,5263,CLOSED

68829,2014-04-22 00:00:00.0,5981,COMPLETE

68830,2014-04-23 00:00:00.0,4952,ON_HOLD

68831,2014-04-24 00:00:00.0,8763,COMPLETE

68832,2014-04-26 00:00:00.0,11203,PROCESSING

68833,2014-04-30 00:00:00.0,12428,PENDING_PAYMENT

68834,2014-05-01 00:00:00.0,6938,COMPLETE

68835,2014-05-02 00:00:00.0,764,COMPLETE

68836,2014-05-03 00:00:00.0,8009,PENDING_PAYMENT

68837,2014-05-07 00:00:00.0,1223,COMPLETE

68838,2014-05-08 00:00:00.0,371,PENDING

68839,2014-05-10 00:00:00.0,10090,COMPLETE

68840,2014-05-11 00:00:00.0,4399,CLOSED

68841,2014-05-14 00:00:00.0,12258,PROCESSING

68842,2014-05-15 00:00:00.0,11268,PENDING

68843,2014-05-16 00:00:00.0,10347,PROCESSING

68844,2014-05-17 00:00:00.0,443,COMPLETE

68845,2014-05-18 00:00:00.0,6584,COMPLETE

68846,2014-05-20 00:00:00.0,3454,PROCESSING

68847,2014-05-21 00:00:00.0,2543,COMPLETE

68848,2014-05-22 00:00:00.0,6517,CLOSED

68849,2014-05-23 00:00:00.0,2356,COMPLETE

68850,2014-05-25 00:00:00.0,8451,COMPLETE

68851,2014-05-26 00:00:00.0,11193,PENDING_PAYMENT

68852,2014-05-29 00:00:00.0,4596,CLOSED

68853,2014-05-31 00:00:00.0,1202,PENDING_PAYMENT

68854,2014-06-01 00:00:00.0,6528,ON_HOLD

68855,2014-06-02 00:00:00.0,403,PENDING_PAYMENT

68856,2014-06-03 00:00:00.0,8853,PROCESSING

68857,2014-06-04 00:00:00.0,312,COMPLETE

68858,2014-06-06 00:00:00.0,10744,COMPLETE

68859,2014-06-11 00:00:00.0,1428,COMPLETE

68860,2014-06-12 00:00:00.0,4229,PENDING

68861,2014-06-13 00:00:00.0,3031,PENDING_PAYMENT

68862,2014-06-15 00:00:00.0,7326,PROCESSING

68863,2014-06-16 00:00:00.0,3361,CLOSED

68864,2014-06-18 00:00:00.0,9634,ON_HOLD

68865,2014-06-19 00:00:00.0,4567,SUSPECTED_FRAUD

68866,2014-06-20 00:00:00.0,3890,PENDING_PAYMENT

68867,2014-06-23 00:00:00.0,869,CANCELED

68868,2014-06-24 00:00:00.0,10184,PENDING

68869,2014-06-25 00:00:00.0,7456,PROCESSING

68870,2014-06-26 00:00:00.0,3343,COMPLETE

68871,2014-06-28 00:00:00.0,4960,PENDING

68872,2014-06-29 00:00:00.0,3354,COMPLETE

68873,2014-06-30 00:00:00.0,4545,PENDING

68874,2014-07-03 00:00:00.0,1601,COMPLETE

68875,2014-07-04 00:00:00.0,10637,ON_HOLD

68876,2014-07-06 00:00:00.0,4124,COMPLETE

68877,2014-07-07 00:00:00.0,9692,ON_HOLD

68878,2014-07-08 00:00:00.0,6753,COMPLETE

68879,2014-07-09 00:00:00.0,778,COMPLETE

68880,2014-07-13 00:00:00.0,1117,COMPLETE

68881,2014-07-19 00:00:00.0,2518,PENDING_PAYMENT

68882,2014-07-22 00:00:00.0,10000,ON_HOLD

68883,2014-07-23 00:00:00.0,5533,COMPLETE

scala> // never use foreach with rdd because spark execute on worker node and collection is required to execute on current node in case of local cluseter it may work

scala> orders.take(5)

res14: Array[String] = Array(1,2013-07-25 00:00:00.0,11599,CLOSED, 2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT, 3,2013-07-25 00:00:00.0,12111,COMPLETE, 4,2013-07-25 00:00:00.0,8827,CLOSED, 5,2013-07-25 00:00:00.0,11318,COMPLETE)

scala> orders.takeSample(true,10)

res15: Array[String] = Array(37752,2014-03-14 00:00:00.0,1535,CLOSED, 11901,2013-10-06 00:00:00.0,6303,PENDING_PAYMENT, 67795,2013-11-09 00:00:00.0,10074,COMPLETE, 46887,2014-05-11 00:00:00.0,6539,COMPLETE, 18113,2013-11-14 00:00:00.0,5855,PENDING, 2168,2013-08-06 00:00:00.0,6316,PROCESSING, 43322,2014-04-20 00:00:00.0,6312,COMPLETE, 58832,2013-09-04 00:00:00.0,8663,COMPLETE, 54474,2014-07-04 00:00:00.0,9280,CANCELED, 46255,2014-05-07 00:00:00.0,5814,COMPLETE)

scala> orders.takeSample(true,10).foreach(println)

42233,2014-04-11 00:00:00.0,4160,COMPLETE

51362,2014-06-12 00:00:00.0,4827,PENDING_PAYMENT

37364,2014-03-12 00:00:00.0,5721,COMPLETE

17984,2013-11-13 00:00:00.0,4306,PENDING

7,2013-07-25 00:00:00.0,4530,COMPLETE

55579,2014-07-11 00:00:00.0,11385,PENDING_PAYMENT

68342,2014-04-13 00:00:00.0,10009,PENDING_PAYMENT

10697,2013-09-28 00:00:00.0,10425,COMPLETE

64216,2014-03-28 00:00:00.0,4750,CLOSED

38671,2014-03-20 00:00:00.0,2549,ON_HOLD

scala> orders.collect

res17: Array[String] = Array(1,2013-07-25 00:00:00.0,11599,CLOSED, 2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT, 3,2013-07-25 00:00:00.0,12111,COMPLETE, 4,2013-07-25

00:00:00.0,8827,CLOSED, 5,2013-07-25 00:00:00.0,11318,COMPLETE, 6,2013-07-25 00:00:00.0,7130,COMPLETE, 7,2013-07-25 00:00:00.0,4530,COMPLETE, 8,2013-07-25 00:00:00.0,2911,PROCESSING, 9,2013-07-25 00:00:00.0,5657,PENDING_PAYMENT, 10,2013-07-25 00:00:00.0,5648,PENDING_PAYMENT, 11,2013-07-25 00:00:00.0,918,PAYMENT_REVIEW, 12,2013-07-25 00:00:00.0,1837,CLOSED, 13,2013-07-25 00:00:00.0,9149,PENDING_PAYMENT, 14,2013-07-25 00:00:00.0,9842,PROCESSING, 15,2013-07-25 00:00:00.0,2568,COMPLETE, 16,2013-07-25 00:00:00.0,7276,PENDING_PAYMENT, 17,2013-07-25 00:00:00.0,2667,COMPLETE, 18,2013-07-25 00:00:00.0,1205,CLOSED, 19,2013-07-25 00:00...

scala> orders.takeOrdered(10,ascending)

<console>:30: error: too many arguments for method takeOrdered: (num: Int)(implicit ord: Ordering[String])Array[String]

orders.takeOrdered(10,ascending)

^

scala> orders.takeOrdered(10)

res19: Array[String] = Array(1,2013-07-25 00:00:00.0,11599,CLOSED, 10,2013-07-25 00:00:00.0,5648,PENDING_PAYMENT, 100,2013-07-25 00:00:00.0,12131,PROCESSING, 1000,2013-07-30 00:00:00.0,2321,CLOSED, 10000,2013-09-25 00:00:00.0,8983,PROCESSING, 10001,2013-09-25 00:00:00.0,316,PENDING_PAYMENT, 10002,2013-09-25 00:00:00.0,1530,COMPLETE, 10003,2013-09-25 00:00:00.0,8099,COMPLETE, 10004,2013-09-25 00:00:00.0,7768,CLOSED, 10005,2013-09-25 00:00:00.0,541,COMPLETE)

scala> sc

res20: org.apache.spark.SparkContext = org.apache.spark.SparkContext@1bb78ae

scala> sqlcontext

<console>:26: error: not found: value sqlcontext

sqlcontext

^

```
scala> sqlContext

res22: org.apache.spark.sql.SQLContext = org.apache.spark.sql.hive.HiveContext@755a4ef5


scala> sqlContext.l

listenerManager   load


scala> sqlContext.load

<console>:26: error: ambiguous reference to overloaded definition,

both method load in class SQLContext of type (source: String, schema:
org.apache.spark.sql.types.StructType, options:
Map[String,String])org.apache.spark.sql.DataFrame

and  method load in class SQLContext of type (source: String, schema:
org.apache.spark.sql.types.StructType, options:
java.util.Map[String,String])org.apache.spark.sql.DataFrame

match expected type ?

        sqlContext.load

                ^


scala> sqlContext.

applySchema          asInstanceOf          baseRelationToDataFrame   cacheTable
clearCache           createDataFrame

createDataset        createExternalTable   dropTempTable         emptyDataFrame
experimental         getAllConfs

getConf              implicits             isCached              isInstanceOf          isRootContext
jdbc
```

jsonFile jsonRDD listenerManager load newSession parquetFile

range read setConf sparkContext sql table

tableNames tables toString udf uncacheTable

scala> sqlContext.read.

asInstanceOf format isInstanceOf jdbc json load option options orc parquet schema

table text toString

scala> sqlContext.read.json("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674")

res24: org.apache.spark.sql.DataFrame = [order_customer_id: bigint, order_date: string, order_id: bigint, order_status: string]

scala> val ordersdf= sqlContext.read.json("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674")

ordersdf: org.apache.spark.sql.DataFrame = [order_customer_id: bigint, order_date: string, order_id: bigint, order_status: string]

scala> ordersdf.show

+----------------+--------------------+--------+---------------+

|order_customer_id| order_date|order_id| order_status|

+----------------+--------------------+--------+---------------+

| 11599|2013-07-25 00:00:...| 1| CLOSED|

| 256|2013-07-25 00:00:...| 2|PENDING_PAYMENT|

| 12111|2013-07-25 00:00:...| 3| COMPLETE|

| 8827|2013-07-25 00:00:...| 4| CLOSED|

| 11318|2013-07-25 00:00:...| 5| COMPLETE|

| 7130|2013-07-25 00:00:...| 6| COMPLETE|

| 4530|2013-07-25 00:00:...| 7| COMPLETE|

| 2911|2013-07-25 00:00:...| 8| PROCESSING|

| 5657|2013-07-25 00:00:...| 9|PENDING_PAYMENT|

| 5648|2013-07-25 00:00:...| 10|PENDING_PAYMENT|

| 918|2013-07-25 00:00:...| 11| PAYMENT_REVIEW|

| 1837|2013-07-25 00:00:...| 12| CLOSED|

| 9149|2013-07-25 00:00:...| 13|PENDING_PAYMENT|

| 9842|2013-07-25 00:00:...| 14| PROCESSING|

| 2568|2013-07-25 00:00:...| 15| COMPLETE|

| 7276|2013-07-25 00:00:...| 16|PENDING_PAYMENT|

| 2667|2013-07-25 00:00:...| 17| COMPLETE|

| 1205|2013-07-25 00:00:...| 18| CLOSED|

| 9488|2013-07-25 00:00:...| 19|PENDING_PAYMENT|

| 9198|2013-07-25 00:00:...| 20| PROCESSING|

+----------+----------+----------+----------+

only showing top 20 rows

scala> sqlContext.load("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674","json")

warning: there were 1 deprecation warning(s); re-run with -deprecation for details

res26: org.apache.spark.sql.DataFrame = [order_customer_id: bigint, order_date: string, order_id: bigint, order_status: string]

scala> sqlContext.load("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674","json").show

warning: there were 1 deprecation warning(s); re-run with -deprecation for details

+----------------+--------------------+--------+---------------+

|order_customer_id| order_date|order_id| order_status|

+----------------+--------------------+--------+---------------+

| 11599|2013-07-25 00:00:...| 1| CLOSED|

| 256|2013-07-25 00:00:...| 2|PENDING_PAYMENT|

| 12111|2013-07-25 00:00:...| 3| COMPLETE|

| 8827|2013-07-25 00:00:...| 4| CLOSED|

| 11318|2013-07-25 00:00:...| 5| COMPLETE|

| 7130|2013-07-25 00:00:...| 6| COMPLETE|

| 4530|2013-07-25 00:00:...| 7| COMPLETE|

| 2911|2013-07-25 00:00:...| 8| PROCESSING|

| 5657|2013-07-25 00:00:...| 9|PENDING_PAYMENT|

| 5648|2013-07-25 00:00:...| 10|PENDING_PAYMENT|

| 918|2013-07-25 00:00:...| 11| PAYMENT_REVIEW|

| 1837|2013-07-25 00:00:...| 12| CLOSED|

| 9149|2013-07-25 00:00:...| 13|PENDING_PAYMENT|

| 9842|2013-07-25 00:00:...| 14| PROCESSING|

| 2568|2013-07-25 00:00:...| 15| COMPLETE|

| 7276|2013-07-25 00:00:...| 16|PENDING_PAYMENT|

```
|            2667|2013-07-25 00:00:...|      17|       COMPLETE|
|            1205|2013-07-25 00:00:...|      18|         CLOSED|
|            9488|2013-07-25 00:00:...|      19|PENDING_PAYMENT|
|            9198|2013-07-25 00:00:...|      20|     PROCESSING|
+----------------+--------------------+--------+---------------+
only showing top 20 rows


scala> sqlContext.load("public/retail_db_json/orders","json").show
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
+----------------+--------------------+--------+---------------+
|order_customer_id|          order_date|order_id|   order_status|
+----------------+--------------------+--------+---------------+
|           11599|2013-07-25 00:00:...|       1|         CLOSED|
|             256|2013-07-25 00:00:...|       2|PENDING_PAYMENT|
|           12111|2013-07-25 00:00:...|       3|       COMPLETE|
|            8827|2013-07-25 00:00:...|       4|         CLOSED|
|           11318|2013-07-25 00:00:...|       5|       COMPLETE|
|            7130|2013-07-25 00:00:...|       6|       COMPLETE|
|            4530|2013-07-25 00:00:...|       7|       COMPLETE|
|            2911|2013-07-25 00:00:...|       8|     PROCESSING|
|            5657|2013-07-25 00:00:...|       9|PENDING_PAYMENT|
|            5648|2013-07-25 00:00:...|      10|PENDING_PAYMENT|
|             918|2013-07-25 00:00:...|      11| PAYMENT_REVIEW|
```

| 1837|2013-07-25 00:00:...| 12| CLOSED|

| 9149|2013-07-25 00:00:...| 13|PENDING_PAYMENT|

| 9842|2013-07-25 00:00:...| 14| PROCESSING|

| 2568|2013-07-25 00:00:...| 15| COMPLETE|

| 7276|2013-07-25 00:00:...| 16|PENDING_PAYMENT|

| 2667|2013-07-25 00:00:...| 17| COMPLETE|

| 1205|2013-07-25 00:00:...| 18| CLOSED|

| 9488|2013-07-25 00:00:...| 19|PENDING_PAYMENT|

| 9198|2013-07-25 00:00:...| 20| PROCESSING|

+----------+--------------------+--------+---------------+

only showing top 20 rows

String and split function:

scala> val orders=sc.textFile("public/retail_db/orders/part-00000")

orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000 MapPartitionsRDD[48] at textFile at <console>:27

scala> orders.first

res29: String = 1,2013-07-25 00:00:00.0,11599,CLOSED

scala> val str = orders.first

str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED

```
scala> str.

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim


scala> str.split
                    def split(String): Array[String]    def split(String, Int): Array[String]


scala> str.split
                    def split(String): Array[String]    def split(String, Int): Array[String]


scala> str.split(",").foreach(println)

1

2013-07-25 00:00:00.0

11599

CLOSED


scala> val a=str.split(",")
```

```
a: Array[String] = Array(1, 2013-07-25 00:00:00.0, 11599, CLOSED)

scala> a(0)

res31: String = 1

scala> a91)

<console>:1: error: ';' expected but ')' found.

      a91)

        ^

scala> a(1)

res32: String = 2013-07-25 00:00:00.0

scala> a(1).contains("2017")

res33: Boolean = false

scala> a(1).contains("2013")

res34: Boolean = true

scala> val orddate = a(1)

orddate: String = 2013-07-25 00:00:00.0

scala> orddate

res35: String = 2013-07-25 00:00:00.0
```

scala> val orderdateonly = orddate.

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim

scala> val orderdateonly = orddate.substring.

asInstanceOf isInstanceOf toString

scala> val orderdateonly = orddate.substring.

asInstanceOf isInstanceOf toString

scala> val orderdateonly = orddate.substring

def substring(Int): String def substring(Int, Int): String

scala> val orderdateonly = orddate.substring

def substring(Int): String def substring(Int, Int): String

```
scala> val orderdateonly = orddate.substring
<console>:35: error: ambiguous reference to overloaded definition,
both method substring in class String of type (x$1: Int, x$2: Int)String
and  method substring in class String of type (x$1: Int)String
match expected type ?
        val orderdateonly = orddate.substring
                            ^

scala> val orderdateonly = orddate.substring(1,10))
<console>:1: error: ';' expected but ')' found.
      val orderdateonly = orddate.substring(1,10))
                                            ^

scala> val orderdateonly = orddate.substring(1,10)
orderdateonly: String = 013-07-25

scala> orddate
res36: String = 2013-07-25 00:00:00.0

scala> val orderdateonly = orddate.substring(0,10)
orderdateonly: String = 2013-07-25

scala> val orderdateonly = orddate.substring(0,9)
orderdateonly: String = 2013-07-2
```

scala> val orderdateonly = orddate.substring(0,10)

orderdateonly: String = 2013-07-25

scala> val orderdateonly = orddate.

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim

scala> val orderdateonly = orddate.replace

replace replaceAll replaceFirst

scala> val orderdateonly = orddate.replace

def replace(Char, Char): String def replace(CharSequence, CharSequence): String

scala> val orderdateonly = orddate.replace

def replace(Char, Char): String def replace(CharSequence, CharSequence): String

```
scala> val orderdateonly = orddate.replace("-","/")

orderdateonly: String = 2013/07/25 00:00:00.0


scala> orderdateonly

res37: String = 2013/07/25 00:00:00.0


scala> orddate.

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim


scala> orddate.indexof

<console>:36: error: value indexof is not a member of String

        orddate.indexof

                ^


scala> orddate.indexof.
```

scala> // never use foreach with rdd because spark execute on worker node and collection is required to execute on current node in case of local cluseter it may work

scala> orders.take(5)

res14: Array[String] = Array(1,2013-07-25 00:00:00.0,11599,CLOSED, 2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT, 3,2013-07-25 00:00:00.0,12111,COMPLETE, 4,2013-07-25 00:00:00.0,8827,CLOSED, 5,2013-07-25 00:00:00.0,11318,COMPLETE)

scala> orders.takeSample(true,10)

res15: Array[String] = Array(37752,2014-03-14 00:00:00.0,1535,CLOSED, 11901,2013-10-06 00:00:00.0,6303,PENDING_PAYMENT, 67795,2013-11-09 00:00:00.0,10074,COMPLETE,

46887,2014-05-11 00:00:00.0,6539,COMPLETE, 18113,2013-11-14 00:00:00.0,5855,PENDING, 2168,2013-08-06 00:00:00.0,6316,PROCESSING, 43322,2014-04-20 00:00:00.0,6312,COMPLETE, 58832,2013-09-04 00:00:00.0,8663,COMPLETE, 54474,2014-07-04 00:00:00.0,9280,CANCELED, 46255,2014-05-07 00:00:00.0,5814,COMPLETE)

scala> orders.takeSample(true,10).foreach(println)

42233,2014-04-11 00:00:00.0,4160,COMPLETE

51362,2014-06-12 00:00:00.0,4827,PENDING_PAYMENT

37364,2014-03-12 00:00:00.0,5721,COMPLETE

17984,2013-11-13 00:00:00.0,4306,PENDING

7,2013-07-25 00:00:00.0,4530,COMPLETE

55579,2014-07-11 00:00:00.0,11385,PENDING_PAYMENT

68342,2014-04-13 00:00:00.0,10009,PENDING_PAYMENT

10697,2013-09-28 00:00:00.0,10425,COMPLETE

64216,2014-03-28 00:00:00.0,4750,CLOSED

38671,2014-03-20 00:00:00.0,2549,ON_HOLD

scala> orders.collect

res17: Array[String] = Array(1,2013-07-25 00:00:00.0,11599,CLOSED, 2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT, 3,2013-07-25 00:00:00.0,12111,COMPLETE, 4,2013-07-25 00:00:00.0,8827,CLOSED, 5,2013-07-25 00:00:00.0,11318,COMPLETE, 6,2013-07-25 00:00:00.0,7130,COMPLETE, 7,2013-07-25 00:00:00.0,4530,COMPLETE, 8,2013-07-25 00:00:00.0,2911,PROCESSING, 9,2013-07-25 00:00:00.0,5657,PENDING_PAYMENT, 10,2013-07-25 00:00:00.0,5648,PENDING_PAYMENT, 11,2013-07-25 00:00:00.0,918,PAYMENT_REVIEW, 12,2013-07-25 00:00:00.0,1837,CLOSED, 13,2013-07-25 00:00:00.0,9149,PENDING_PAYMENT, 14,2013-07-25 00:00:00.0,9842,PROCESSING, 15,2013-07-25 00:00:00.0,2568,COMPLETE, 16,2013-07-25 00:00:00.0,7276,PENDING_PAYMENT, 17,2013-07-25 00:00:00.0,2667,COMPLETE, 18,2013-07-25 00:00:00.0,1205,CLOSED, 19,2013-07-25 00:00...

scala> orders.takeOrdered(10,ascending)

<console>:30: error: too many arguments for method takeOrdered: (num: Int)(implicit ord: Ordering[String])Array[String]

orders.takeOrdered(10,ascending)

^

scala> orders.takeOrdered(10)

res19: Array[String] = Array(1,2013-07-25 00:00:00.0,11599,CLOSED, 10,2013-07-25 00:00:00.0,5648,PENDING_PAYMENT, 100,2013-07-25 00:00:00.0,12131,PROCESSING, 1000,2013-07-30 00:00:00.0,2321,CLOSED, 10000,2013-09-25 00:00:00.0,8983,PROCESSING, 10001,2013-09-25 00:00:00.0,316,PENDING_PAYMENT, 10002,2013-09-25 00:00:00.0,1530,COMPLETE, 10003,2013-09-25 00:00:00.0,8099,COMPLETE, 10004,2013-09-25 00:00:00.0,7768,CLOSED, 10005,2013-09-25 00:00:00.0,541,COMPLETE)

scala> sc

res20: org.apache.spark.SparkContext = org.apache.spark.SparkContext@1bb78ae

scala> sqlcontext

<console>:26: error: not found: value sqlcontext

sqlcontext

^

scala> sqlContext

res22: org.apache.spark.sql.SQLContext = org.apache.spark.sql.hive.HiveContext@755a4ef5

scala> sqlContext.l

listenerManager load

scala> sqlContext.load

<console>:26: error: ambiguous reference to overloaded definition,

both method load in class SQLContext of type (source: String, schema: org.apache.spark.sql.types.StructType, options: Map[String,String])org.apache.spark.sql.DataFrame

and method load in class SQLContext of type (source: String, schema: org.apache.spark.sql.types.StructType, options: java.util.Map[String,String])org.apache.spark.sql.DataFrame

match expected type ?

sqlContext.load

^

scala> sqlContext.

applySchema asInstanceOf baseRelationToDataFrame cacheTable clearCache createDataFrame

createDataset createExternalTable dropTempTable emptyDataFrame experimental getAllConfs

getConf implicits isCached isInstanceOf isRootContext jdbc

jsonFile jsonRDD listenerManager load newSession parquetFile

range read setConf sparkContext sql table

tableNames tables toString udf uncacheTable

scala> sqlContext.read.

asInstanceOf format isInstanceOf jdbc json load option options orc parquet schema

table text toString

```
scala> sqlContext.read.json("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674")

res24: org.apache.spark.sql.DataFrame = [order_customer_id: bigint, order_date: string, order_id: bigint, order_status: string]

scala> val ordersdf= sqlContext.read.json("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674")

ordersdf: org.apache.spark.sql.DataFrame = [order_customer_id: bigint, order_date: string, order_id: bigint, order_status: string]

scala> ordersdf.show

+-----------------+--------------------+--------+---------------+

|order_customer_id|          order_date|order_id|   order_status|

+-----------------+--------------------+--------+---------------+

|            11599|2013-07-25 00:00:...|       1|         CLOSED|

|              256|2013-07-25 00:00:...|       2|PENDING_PAYMENT|

|            12111|2013-07-25 00:00:...|       3|       COMPLETE|

|             8827|2013-07-25 00:00:...|       4|         CLOSED|

|            11318|2013-07-25 00:00:...|       5|       COMPLETE|

|             7130|2013-07-25 00:00:...|       6|       COMPLETE|

|             4530|2013-07-25 00:00:...|       7|       COMPLETE|

|             2911|2013-07-25 00:00:...|       8|     PROCESSING|
```

```
|            5657|2013-07-25 00:00:...|       9|PENDING_PAYMENT|
|            5648|2013-07-25 00:00:...|      10|PENDING_PAYMENT|
|             918|2013-07-25 00:00:...|      11| PAYMENT_REVIEW|
|            1837|2013-07-25 00:00:...|      12|         CLOSED|
|            9149|2013-07-25 00:00:...|      13|PENDING_PAYMENT|
|            9842|2013-07-25 00:00:...|      14|     PROCESSING|
|            2568|2013-07-25 00:00:...|      15|       COMPLETE|
|            7276|2013-07-25 00:00:...|      16|PENDING_PAYMENT|
|            2667|2013-07-25 00:00:...|      17|       COMPLETE|
|            1205|2013-07-25 00:00:...|      18|         CLOSED|
|            9488|2013-07-25 00:00:...|      19|PENDING_PAYMENT|
|            9198|2013-07-25 00:00:...|      20|     PROCESSING|
+-----------------+--------------------+--------+---------------+
only showing top 20 rows
```

```
scala> sqlContext.load("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674","json")
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
res26: org.apache.spark.sql.DataFrame = [order_customer_id: bigint, order_date: string, order_id: bigint, order_status: string]
```

```
scala> sqlContext.load("public/retail_db_json/orders/part-r-00000-990f5773-9005-49ba-b670-631286032674","json").show
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
```

order_customer_id	order_date	order_id	order_status
11599	2013-07-25 00:00:...	1	CLOSED
256	2013-07-25 00:00:...	2	PENDING_PAYMENT
12111	2013-07-25 00:00:...	3	COMPLETE
8827	2013-07-25 00:00:...	4	CLOSED
11318	2013-07-25 00:00:...	5	COMPLETE
7130	2013-07-25 00:00:...	6	COMPLETE
4530	2013-07-25 00:00:...	7	COMPLETE
2911	2013-07-25 00:00:...	8	PROCESSING
5657	2013-07-25 00:00:...	9	PENDING_PAYMENT
5648	2013-07-25 00:00:...	10	PENDING_PAYMENT
918	2013-07-25 00:00:...	11	PAYMENT_REVIEW
1837	2013-07-25 00:00:...	12	CLOSED
9149	2013-07-25 00:00:...	13	PENDING_PAYMENT
9842	2013-07-25 00:00:...	14	PROCESSING
2568	2013-07-25 00:00:...	15	COMPLETE
7276	2013-07-25 00:00:...	16	PENDING_PAYMENT
2667	2013-07-25 00:00:...	17	COMPLETE
1205	2013-07-25 00:00:...	18	CLOSED
9488	2013-07-25 00:00:...	19	PENDING_PAYMENT
9198	2013-07-25 00:00:...	20	PROCESSING

only showing top 20 rows

```
scala> sqlContext.load("public/retail_db_json/orders","json").show
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
+-----------------+--------------------+--------+---------------+
|order_customer_id|          order_date|order_id|   order_status|
+-----------------+--------------------+--------+---------------+
|            11599|2013-07-25 00:00:...|       1|         CLOSED|
|              256|2013-07-25 00:00:...|       2|PENDING_PAYMENT|
|            12111|2013-07-25 00:00:...|       3|       COMPLETE|
|             8827|2013-07-25 00:00:...|       4|         CLOSED|
|            11318|2013-07-25 00:00:...|       5|       COMPLETE|
|             7130|2013-07-25 00:00:...|       6|       COMPLETE|
|             4530|2013-07-25 00:00:...|       7|       COMPLETE|
|             2911|2013-07-25 00:00:...|       8|     PROCESSING|
|             5657|2013-07-25 00:00:...|       9|PENDING_PAYMENT|
|             5648|2013-07-25 00:00:...|      10|PENDING_PAYMENT|
|              918|2013-07-25 00:00:...|      11| PAYMENT_REVIEW|
|             1837|2013-07-25 00:00:...|      12|         CLOSED|
|             9149|2013-07-25 00:00:...|      13|PENDING_PAYMENT|
|             9842|2013-07-25 00:00:...|      14|     PROCESSING|
|             2568|2013-07-25 00:00:...|      15|       COMPLETE|
|             7276|2013-07-25 00:00:...|      16|PENDING_PAYMENT|
```

```
|          2667|2013-07-25 00:00:...|      17|       COMPLETE|
|          1205|2013-07-25 00:00:...|      18|         CLOSED|
|          9488|2013-07-25 00:00:...|      19|PENDING_PAYMENT|
|          9198|2013-07-25 00:00:...|      20|     PROCESSING|
+--------------+--------------------+--------+---------------+
only showing top 20 rows
```

```
scala> val orders=sc.textFile("public/retail_db/orders/part-00000")
orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000 MapPartitionsRDD[48] at textFile at <console>:27

scala> orders.first
res29: String = 1,2013-07-25 00:00:00.0,11599,CLOSED

scala> val str = orders.first
str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED

scala> str.
+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount
codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith
equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf
```

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim

scala> str.split

def split(String): Array[String] def split(String, Int): Array[String]

scala> str.split

def split(String): Array[String] def split(String, Int): Array[String]

scala> str.split(",").foreach(println)

1

2013-07-25 00:00:00.0

11599

CLOSED

scala> val a=str.split(",")

a: Array[String] = Array(1, 2013-07-25 00:00:00.0, 11599, CLOSED)

scala> a(0)

res31: String = 1

scala> a91)

<console>:1: error: ';' expected but ')' found.

a91)

^

scala> a(1)

res32: String = 2013-07-25 00:00:00.0

scala> a(1).contains("2017")

res33: Boolean = false

scala> a(1).contains("2013")

res34: Boolean = true

scala> val orddate = a(1)

orddate: String = 2013-07-25 00:00:00.0

scala> orddate

res35: String = 2013-07-25 00:00:00.0

scala> val orderdateonly = orddate.

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim

scala> val orderdateonly = orddate.substring.

asInstanceOf isInstanceOf toString

scala> val orderdateonly = orddate.substring.

asInstanceOf isInstanceOf toString

scala> val orderdateonly = orddate.substring

def substring(Int): String def substring(Int, Int): String

scala> val orderdateonly = orddate.substring

def substring(Int): String def substring(Int, Int): String

scala> val orderdateonly = orddate.substring

<console>:35: error: ambiguous reference to overloaded definition,

both method substring in class String of type (x$1: Int, x$2: Int)String

and method substring in class String of type (x$1: Int)String

match expected type ?

```
        val orderdateonly = orddate.substring
                                  ^

scala> val orderdateonly = orddate.substring(1,10))
<console>:1: error: ';' expected but ')' found.
      val orderdateonly = orddate.substring(1,10))
                                              ^

scala> val orderdateonly = orddate.substring(1,10)
orderdateonly: String = 013-07-25

scala> orddate
res36: String = 2013-07-25 00:00:00.0

scala> val orderdateonly = orddate.substring(0,10)
orderdateonly: String = 2013-07-25

scala> val orderdateonly = orddate.substring(0,9)
orderdateonly: String = 2013-07-2

scala> val orderdateonly = orddate.substring(0,10)
orderdateonly: String = 2013-07-25

scala> val orderdateonly = orddate.
```

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll

replaceFirst split startsWith subSequence substring toCharArray toLowerCase

toString toUpperCase trim

scala> val orderdateonly = orddate.replace

replace replaceAll replaceFirst

scala> val orderdateonly = orddate.replace

def replace(Char, Char): String def replace(CharSequence, CharSequence): String

scala> val orderdateonly = orddate.replace

def replace(Char, Char): String def replace(CharSequence, CharSequence): String

scala> val orderdateonly = orddate.replace("-","/")

orderdateonly: String = 2013/07/25 00:00:00.0

scala> orderdateonly

```
res37: String = 2013/07/25 00:00:00.0


scala> orddate.

+ asInstanceOf charAt chars codePointAt
codePointBefore codePointCount

codePoints compareTo compareToIgnoreCase concat contains
contentEquals endsWith

equalsIgnoreCase getBytes getChars indexOf intern isEmpty
isInstanceOf

lastIndexOf length matches offsetByCodePoints regionMatches
replace replaceAll

replaceFirst split startsWith subSequence substring
toCharArray toLowerCase

toString toUpperCase trim


scala> orddate.indexof

<console>:36: error: value indexof is not a member of String

        orddate.indexof

               ^


scala> orddate.indexof.(

<console>:1: error: identifier expected but '(' found.

    orddate.indexof.(

               ^


scala> val orders=sc.textFile("public/retail_db/orders/part-00000")
```

```
orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000
MapPartitionsRDD[50] at textFile at <console>:27


scala> orders.first

res39: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> val orderdate=orders.split(",")(2)

<console>:29: error: value split is not a member of org.apache.spark.rdd.RDD[String]

       val orderdate=orders.split(",")(2)

                            ^


scala> val orderdate=orders.split(",").<console>:1: error: unclosed string literal

")

^


    | .

<console>:2: error: identifier expected but '.' found.

      .

      ^


scala> val order_array=orders.split(",")

<console>:29: error: value split is not a member of org.apache.spark.rdd.RDD[String]

       val order_array=orders.split(",")

                            ^
```

```
scala> val orders=sc.textFile("public/retail_db/orders/part-00000")

orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000
MapPartitionsRDD[52] at textFile at <console>:27


scala> val order_array=orders.split(",")

<console>:29: error: value split is not a member of org.apache.spark.rdd.RDD[String]

       val order_array=orders.split(",")

                           ^


scala> orders

res40: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000
MapPartitionsRDD[52] at textFile at <console>:27


scala> orders.first

res41: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> val str=orders.first

str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> val date=str.split(",")(1)

date: String = 2013-07-25 00:00:00.0


scala> date

res42: String = 2013-07-25 00:00:00.0
```

```
scala> val date=str.split(",")(1).substring(0,10)

date: String = 2013-07-25


scala> val date=str.split(",")(1).substring(0,10).replace("-","")

date: String = 20130725


scala> val date=str.split(",")(1).substring(0,10).replace("-","").toInt

date: Int = 20130725


scala> orders

res43: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000
MapPartitionsRDD[52] at textFile at <console>:27


scala> orders.map

map                mapPartitions          mapPartitionsWithContext   mapPartitionsWithIndex
mapPartitionsWithSplit     mapWith


scala> orders.map

                                          def map[U](f: T => U)(implicit scala.reflect.ClassTag[U]):
RDD[U]


scala> orders.ma

map                mapPartitions          mapPartitionsWithContext   mapPartitionsWithIndex
mapPartitionsWithSplit     mapWith

max
```

```
scala> orders.ma

map                mapPartitions          mapPartitionsWithContext   mapPartitionsWithIndex
mapPartitionsWithSplit     mapWith

max


scala> orders.ma

map                mapPartitions          mapPartitionsWithContext   mapPartitionsWithIndex
mapPartitionsWithSplit     mapWith

max


scala> orders.map

map                mapPartitions          mapPartitionsWithContext   mapPartitionsWithIndex
mapPartitionsWithSplit     mapWith


scala> orders.map

                                          def map[U](f: T => U)(implicit scala.reflect.ClassTag[U]):
RDD[U]


scala> orders.map

                                          def map[U](f: T => U)(implicit scala.reflect.ClassTag[U]):
RDD[U]


scala> orders.map

                                          def map[U](f: T => U)(implicit scala.reflect.ClassTag[U]):
RDD[U]
```

```
scala> val date=str.split(",")(1).substring(0,10).replace("-","").toInt

date: Int = 20130725


scala> val orderDates = orders.map((str :String) => {str.split(",")
(1).substring(0,10).replace("-","").toInt})

orderDates: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[53] at map at <console>:31


scala> orderDates

res44: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[53] at map at <console>:31


scala> orderDates.take(10)

res45: Array[Int] = Array(20130725, 20130725, 20130725, 20130725, 20130725, 20130725,
20130725, 20130725, 20130725, 20130725)


scala> orderDates.take(10).foreach(println)

20130725

20130725

20130725

20130725

20130725

20130725

20130725

20130725

20130725

20130725
```

```
scala> val ordersPairedRDD = orders.map(order  => {
     |              val o =  order.split(",")
     |              (o(0).toInt, o(1).substring(0,10).replace("-","").toInt) })
ordersPairedRDD: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[56] at map at
<console>:29


scala> ordersPairedRDD.take(10).foreach(println)
(1,20130725)
(2,20130725)
(3,20130725)
(4,20130725)
(5,20130725)
(6,20130725)
(7,20130725)
(8,20130725)
(9,20130725)
(10,20130725)
```

```
scala> val orderItems=sc.textFile("public/retail_db/order_items")

orderItems: org.apache.spark.rdd.RDD[String] = public/retail_db/order_items
MapPartitionsRDD[60] at textFile at <console>:27


scala> orderItems.first

res49: String = 1,1,957,1,299.98,299.98


scala> val orderItemssPairedRDD = orderItems.map(orderItem  => {

    |    (orderItem.split(",")(1).toInt,orderItem)

    | })

orderItemssPairedRDD: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[61] at map
at <console>:29


scala> orderItemssPairedRDD.take(10).foreach(println)

(1,1,1,957,1,299.98,299.98)

(2,2,2,1073,1,199.99,199.99)

(2,3,2,502,5,250.0,50.0)

(2,4,2,403,1,129.99,129.99)

(4,5,4,897,2,49.98,24.99)

(4,6,4,365,5,299.95,59.99)

(4,7,4,502,3,150.0,50.0)

(4,8,4,1014,4,199.92,49.98)

(5,9,5,957,1,299.98,299.98)

(5,10,5,365,5,299.95,59.99)
```

Map:

```
scala> val orders=sc.textFile("public/retail_db/orders/part-00000")

orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000
MapPartitionsRDD[7] at textFile at <console>:27

scala> val str= orders.first

str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toLowerCase } )

orderstatus: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[8] at map at <console>:29

scala> orderstatus.take(10).foreach(println)

closed

pending_payment

complete

closed

complete

complete

complete

processing

pending_payment
```

pending_payment

wordcount:

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/
```

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_60)

Type in expressions to have them evaluated.

Type :help for more information.

17/11/27 09:43:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

17/11/27 09:43:10 WARN util.Utils: Your hostname, localhost resolves to a loopback address: 127.0.0.1; using 192.168.112.160 instead (on interface eth3)

17/11/27 09:43:10 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address

17/11/27 09:43:25 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

Spark context available as sc (master = local[*], app id = local-1511804596167).

SQL context available as sqlContext.

scala> val orders=sc.textFile("public/retail_db/orders/part-00000")

orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000 MapPartitionsRDD[1] at textFile at <console>:27

```
scala> val str= orders.first

str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> val orderstatus=orders.map(ordstatus => (order.split(",")(3).toLower  )  )

<console>:29: error: not found: value order

       val orderstatus=orders.map(ordstatus => (order.split(",")(3).toLower  )  )

                                  ^


scala> val orderstatus=orders.map(ordstatus => (orderstatus.split(",")(3).toLower  )  )

<console>:29: error: recursive value orderstatus needs type

       val orderstatus=orders.map(ordstatus => (orderstatus.split(",")(3).toLower  )  )

                                  ^

<console>:29: error: value split is not a member of org.apache.spark.rdd.RDD[Nothing]

       val orderstatus=orders.map(ordstatus => (orderstatus.split(",")(3).toLower  )  )

                                           ^


scala> val orderstatus=orders.map(ordstatus => {orderstatus.split(",")(3).toLower } )

<console>:29: error: recursive value orderstatus needs type

       val orderstatus=orders.map(ordstatus => {orderstatus.split(",")(3).toLower } )

                                  ^

<console>:29: error: value split is not a member of org.apache.spark.rdd.RDD[Nothing]

       val orderstatus=orders.map(ordstatus => {orderstatus.split(",")(3).toLower } )

                                           ^
```

```
scala> val orderstatus=orders.map(ordstatus => {orderstatus.split(",")(3) } )
<console>:29: error: recursive value orderstatus needs type
       val orderstatus=orders.map(ordstatus => {orderstatus.split(",")(3) } )
                                  ^
<console>:29: error: value split is not a member of org.apache.spark.rdd.RDD[Nothing]
       val orderstatus=orders.map(ordstatus => {orderstatus.split(",")(3) } )
                                           ^

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3) } )
orderstatus: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at map at <console>:29

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toLower } )
<console>:29: error: value toLower is not a member of String
       val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toLower } )
                                                          ^

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).<console>:1: error:
unclosed string literal
")(3)
^
<console>:1: error: unclosed string literal
")(3)
^
<console>:1: error: unclosed string literal
")(3)
```

^

<console>:1: error: identifier expected but '}' found.

val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3). })

^

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3)})

orderstatus: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at map at <console>:29

scala> orderstatus.take(10)

res0: Array[String] = Array(CLOSED, PENDING_PAYMENT, COMPLETE, CLOSED, COMPLETE, COMPLETE, COMPLETE, PROCESSING, PENDING_PAYMENT, PENDING_PAYMENT)

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toUpper})

<console>:29: error: value toUpper is not a member of String

val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toUpper})

^

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).substrin(0,4)})

<console>:29: error: value substrin is not a member of String

val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).substrin(0,4)})

^

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).substring(0,4)})

orderstatus: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at map at <console>:29

```
scala> orderstatus.take(10)

res1: Array[String] = Array(CLOS, PEND, COMP, CLOS, COMP, COMP, COMP, PROC, PEND, PEND)


scala> orderstatus.

++                  aggregate           asInstanceOf        cache               cartesian
checkpoint          coalesce            collect

compute             context             count               countApprox
countApproxDistinct countByValue        countByValueApprox  dependencies

distinct            filter              filterWith          first               flatMap
flatMapWith         fold                foreach

foreachPartition    foreachWith         getCheckpointFile   getNumPartitions
getStorageLevel     glom                groupBy             id

intersection        isCheckpointed      isEmpty             isInstanceOf        iterator
keyBy               localCheckpoint     map

mapPartitions       mapPartitionsWithContext  mapPartitionsWithIndex
mapPartitionsWithSplit  mapWith         max                 min                 name

name_=              partitioner         partitions          persist             pipe
preferredLocations  randomSplit         reduce

repartition         sample              saveAsObjectFile    saveAsTextFile      setName
sortBy              sparkContext        subtract

take                takeOrdered         takeSample          toArray             toDebugString
toJavaRDD           toLocalIterator     toString

top                 treeAggregate       treeReduce          union               unpersist
zip                 zipPartitions       zipWithIndex

zipWithUniqueId


scala> orderstatus.
```

++ aggregate asInstanceOf cache cartesian checkpoint coalesce collect

compute context count countApprox countApproxDistinct countByValue countByValueApprox dependencies

distinct filter filterWith first flatMap flatMapWith fold foreach

foreachPartition foreachWith getCheckpointFile getNumPartitions getStorageLevel glom groupBy id

intersection isCheckpointed isEmpty isInstanceOf iterator keyBy localCheckpoint map

mapPartitions mapPartitionsWithContext mapPartitionsWithIndex mapPartitionsWithSplit mapWith max min name

name_= partitioner partitions persist pipe preferredLocations randomSplit reduce

repartition sample saveAsObjectFile saveAsTextFile setName sortBy sparkContext subtract

take takeOrdered takeSample toArray toDebugString toJavaRDD toLocalIterator toString

top treeAggregate treeReduce union unpersist zip zipPartitions zipWithIndex

zipWithUniqueId

scala> val str = orders.first

str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED

scala> str.

+ asInstanceOf charAt chars codePointAt codePointBefore codePointCount codePoints compareTo compareToIgnoreCase

concat contains contentEquals endsWith equalsIgnoreCase getBytes getChars indexOf intern isEmpty

isInstanceOf lastIndexOf length matches offsetByCodePoints regionMatches replace replaceAll replaceFirst split

startsWith subSequence substring toCharArray toLowerCase toString toUpperCase trim

scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toLowerCase })

orderstatus: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[5] at map at <console>:29

scala> orderstatus.take(10)

res2: Array[String] = Array(closed, pending_payment, complete, closed, complete, complete, complete, processing, pending_payment, pending_payment)

scala> orderstatus.take(10).foreach(println)

closed

pending_payment

complete

closed

complete

complete

complete

processing

pending_payment

pending_payment

```
scala> val orders=sc.textFile("public/retail_db/orders/part-00000")

orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000
MapPartitionsRDD[7] at textFile at <console>:27


scala> val str= orders.first

str: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> val orderstatus=orders.map(ordstatus => {ordstatus.split(",")(3).toLowerCase } )

orderstatus: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[8] at map at <console>:29


scala> orderstatus.take(10).foreach(println)

closed

pending_payment

complete

closed

complete

complete

complete

processing

pending_payment

pending_payment


scala> val l=List("Hello","How are you","Let us perform word count","As part of the count
program")
```

l: List[String] = List(Hello, How are you, Let us perform word count, As part of the count program)

scala> l.collect.foreach(println)

<console>:28: error: missing arguments for method collect in trait TraversableLike;

follow this method with `_' if you want to treat it as a partially applied function

l.collect.foreach(println)

^

scala> val l_rdd = l.parallelize(l)

<console>:27: error: value parallelize is not a member of List[String]

val l_rdd = l.parallelize(l)

^

scala> val l=List("Hello","How are you","Let us perform word count","As part of the count program")

l: List[String] = List(Hello, How are you, Let us perform word count, As part of the count program)

scala> val l_rdd = sc.parallelize(l)

l_rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[9] at parallelize at <console>:29

scala> l_rdd.collect.foreach(println)

Hello

How are you

Let us perform word count

As part of the count program

```
scala> val l_map = l_rdd.map(ele => ele.split(" "))

l_map: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[10] at map at
<console>:31


scala> val l_flatmap = l_rdd.flatmap(ele => ele.split(" "))

<console>:31: error: value flatmap is not a member of org.apache.spark.rdd.RDD[String]

       val l_flatmap = l_rdd.flatmap(ele => ele.split(" "))

                       ^


scala> val l_flatmap = l_rdd.flatMap(ele => ele.split(" "))

l_flatmap: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[11] at flatMap at
<console>:31


scala> l_map.collect.foreach(println)

[Ljava.lang.String;@70f0e76d

[Ljava.lang.String;@7d877489

[Ljava.lang.String;@1e575053

[Ljava.lang.String;@13bbc985


scala> l_flatmap.collect.foreach(println)

Hello

How
```

are

you

Let

us

perform

word

count

As

part

of

the

count

program

```
scala> val wordcount = l_flatmap.map(word => (word,"")).countByKey

wordcount: scala.collection.Map[String,Long] = Map(program -> 1, count -> 2, are -> 1, How -> 1, Let -> 1, us -> 1, you -> 1, Hello -> 1, perform -> 1, part -> 1, As -> 1, word -> 1, of -> 1, the -> 1)

scala> wordcount.take(10).foreach(println)

(program,1)

(count,2)

(are,1)

(How,1)

(Let,1)

(us,1)

(you,1)
```

(Hello,1)

(perform,1)

(part,1)

scala>

Filter:

a> val orders=sc.textFile("public/retail_db/orders/part-00000")

orders: org.apache.spark.rdd.RDD[String] = public/retail_db/orders/part-00000 MapPartitionsRDD[1] at textFile at <console>:27

scala> orders.take(10).foreach(println)

1,2013-07-25 00:00:00.0,11599,CLOSED

2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT

3,2013-07-25 00:00:00.0,12111,COMPLETE

4,2013-07-25 00:00:00.0,8827,CLOSED

5,2013-07-25 00:00:00.0,11318,COMPLETE

6,2013-07-25 00:00:00.0,7130,COMPLETE

7,2013-07-25 00:00:00.0,4530,COMPLETE

8,2013-07-25 00:00:00.0,2911,PROCESSING

9,2013-07-25 00:00:00.0,5657,PENDING_PAYMENT

10,2013-07-25 00:00:00.0,5648,PENDING_PAYMENT

scala> orders.filter.

asInstanceOf isInstanceOf toString

scala> orders.filter.

asInstanceOf isInstanceOf toString

scala> orders.filter

filter filterWith

scala> orders.filter

def filter(f: T => Boolean): RDD[T]

scala> val orders_completed= orders.filter({order_comp => order_comp.split(",")(3)==true})

orders_completed: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:29

scala> orders_completed.take(10).foreach(println)

scala> val orders_completed= orders.filter({order_comp => order_comp.split(",")(3)== "COMPLETE"})

orders_completed: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at filter at <console>:29

scala> orders_completed.take(10).foreach(println)

3,2013-07-25 00:00:00.0,12111,COMPLETE

5,2013-07-25 00:00:00.0,11318,COMPLETE

6,2013-07-25 00:00:00.0,7130,COMPLETE

7,2013-07-25 00:00:00.0,4530,COMPLETE

15,2013-07-25 00:00:00.0,2568,COMPLETE

17,2013-07-25 00:00:00.0,2667,COMPLETE

22,2013-07-25 00:00:00.0,333,COMPLETE

26,2013-07-25 00:00:00.0,7562,COMPLETE

28,2013-07-25 00:00:00.0,656,COMPLETE

32,2013-07-25 00:00:00.0,3960,COMPLETE

scala> orders.count

res3: Long = 68883

scala> orders.filter({order_comp => order_comp.split(",")(3)==
"COMPLETE"}).take(10).foreach(println)

3,2013-07-25 00:00:00.0,12111,COMPLETE

5,2013-07-25 00:00:00.0,11318,COMPLETE

6,2013-07-25 00:00:00.0,7130,COMPLETE

7,2013-07-25 00:00:00.0,4530,COMPLETE

15,2013-07-25 00:00:00.0,2568,COMPLETE

17,2013-07-25 00:00:00.0,2667,COMPLETE

22,2013-07-25 00:00:00.0,333,COMPLETE

26,2013-07-25 00:00:00.0,7562,COMPLETE

28,2013-07-25 00:00:00.0,656,COMPLETE

32,2013-07-25 00:00:00.0,3960,COMPLETE

```
scala> orders.filter({order_comp => order_comp.split(",")(3)== "COMPLETE"}).count

res5: Long = 22899


scala> val s = orders.first

s: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> s.contain("2013")

<console>:32: error: value contain is not a member of String

              s.contain("2013")

                ^


scala> s.contains("COMPLETE")||s.contains("CLOSED")

res7: Boolean = true


scala> s.split(",")(3)=="COMPLETE"||s.split(",")(3)=="CLOSED"

res8: Boolean = true


scala> s

res9: String = 1,2013-07-25 00:00:00.0,11599,CLOSED


scala> (s.split(",")(3)=="COMPLETE"||s.split(",")(3)=="CLOSED") && (s.splt(",")
(1).contains("2013-07-25")

     | )

<console>:32: error: value splt is not a member of String
```

```
        (s.split(",")(3)=="COMPLETE"||s.split(",")(3)=="CLOSED") && (s.splt(",")
(1).contains("2013-07-25")
                                                    ^

scala> (s.split(",")(3)=="COMPLETE"||s.split(",")(3)=="CLOSED") && (s.split(",")
(1).contains("2013-07-25"))
res11: Boolean = true

scala> orders.map(order => order.split(",")(3)).distinct
res12: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[9] at distinct at <console>:30

scala> orders.map(order => order.split(",")(3)).distinct.collect.foreach(println)
PENDING_PAYMENT
CANCELED
CLOSED
PAYMENT_REVIEW
PENDING
ON_HOLD
PROCESSING
SUSPECTED_FRAUD
COMPLETE

scala> val ordersfiltered = orders.filter( order => {
     |                 val o= order.split(",")
     |       (o(3) == "COMPLETE" || O(3) == "CLOSED" ) && (O(1).contains("2013-09"))
```

```
    | }
    | )
<console>:31: error: not found: value O
            (o(3) == "COMPLETE" || O(3) == "CLOSED" ) && (O(1).contains("2013-09"))
                                   ^


scala> val ordersfiltered = orders.filter( order => {
    |                   val o= order.split(",")
    |         (o(3) == "COMPLETE" || o(3) == "CLOSED" ) && (O(1).contains("2013-09"))
    | }
    | )
<console>:31: error: not found: value O
            (o(3) == "COMPLETE" || o(3) == "CLOSED" ) && (O(1).contains("2013-09"))
                                                  ^


scala> val ordersfiltered = orders.filter( order => {
    |                   val o= order.split(",")
    |         (o(3) == "COMPLETE" || o(3) == "CLOSED" ) && (o(1).contains("2013-09"))
    | }
    | )
ordersfiltered: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[14] at filter at
<console>:29


scala> ordersfiltered.take(10).foreach(println)
6059,2013-09-01 00:00:00.0,11516,COMPLETE
```

6061,2013-09-01 00:00:00.0,7209,COMPLETE

6063,2013-09-01 00:00:00.0,9236,CLOSED

6065,2013-09-01 00:00:00.0,2114,COMPLETE

6066,2013-09-01 00:00:00.0,5068,COMPLETE

6069,2013-09-01 00:00:00.0,4242,COMPLETE

6075,2013-09-01 00:00:00.0,9496,COMPLETE

6076,2013-09-01 00:00:00.0,7838,COMPLETE

6077,2013-09-01 00:00:00.0,9119,CLOSED

6078,2013-09-01 00:00:00.0,10377,CLOSED

scala>

Join:

```
//count of orders for each status
val orders=sc.textFile("public/retail_db/orders/part-00000")
//we can have key as 1 or null or 2 doesn't matter
orders.map(order => order.split("")(3)("1")).countByKey("").foreach(println)



//total revenue
val orderItems=sc.textFile("public/retail_db/order_items")
val orderItemsRevenue = orderItems.map(revenue => revenue.split(",")(4).toFloat)
orderItemsRevenue.reduce((total,revenue) => total+revenue)



//max revnue of an item
```

```
val orderItemsMaxRevenue = orderItemsRevenue.reduce((max,revenue) => {
if (max > revenue) max else revenue
})
```

```
//aggregation
val orderItems=sc.textFile("public/retail_db/order_items")
//get revenue per order_id
//get data in descending order by order_item_subtotal for each order_id

val orderItemsMap = orderItems.
map(oi => (oi.split(",")(1).toInt,oi.split(",")(4).toFloat))

val orderItemGBK = orderItemsMap.groupByKey
//Get revenue per order_id
orderItemGBK.map( rec => (rec._1, rec._2.toList.sum)).take(10).foreach(println)

// Get data in descending order by order_item_subtotal for each order_id

val ordersSortedbyRevenue =
orderItemGBK.
flatMap(rec => {
```

```
        rec._2.toList.sortBy(o => -o).map(k => (rec._1,k))
  })

ordersSortedbyRevenue.take(10).foreach(println)
```

```
scala> val orderItems=sc.textFile("public/retail_db/order_items")
orderItems: org.apache.spark.rdd.RDD[String] = public/retail_db/order_items
MapPartitionsRDD[6] at textFile at <console>:27

scala> val orderItemsMap = orderItems.
     | map(oi => (oi.split(",")(1).toInt,oi.split(",")(4).toFloat))
orderItemsMap: org.apache.spark.rdd.RDD[(Int, Float)] = MapPartitionsRDD[7] at map at
<console>:30

scala> orderItemsMap.take(10).foreach(println)
(1,299.98)
(2,199.99)
(2,250.0)
(2,129.99)
(4,49.98)
(4,299.95)
(4,150.0)
(4,199.92)
(5,299.98)
```

(5,299.95)

scala> val revenuePerOrderId = orderItemsMap.reduceByKey((total,revenue) => total+ revenue)

revenuePerOrderId: org.apache.spark.rdd.RDD[(Int, Float)] = ShuffledRDD[8] at reduceByKey at <console>:31

scala> revenuePerOrderId.take(10).foreach(println)

(65722,1319.8899)

(68522,329.99)

(23776,329.98)

(34207,219.94)

(53499,284.88998)

(32676,719.91003)

(53926,219.97)

(4926,939.85)

(38926,1049.9)

(51620,999.85004)

scala> val minRevenuePerOrderId = orderItemsMap.reduceByKey((min,revenue) => if (min,revenue) revenue else min)

<console>:1: error: ')' expected but ',' found.

    val minRevenuePerOrderId = orderItemsMap.reduceByKey((min,revenue) => if (min,revenue) revenue else min)

^

scala> val minRevenuePerOrderId = orderItemsMap.reduceByKey((min,revenue) => if (min<revenue) revenue else min)

minRevenuePerOrderId: org.apache.spark.rdd.RDD[(Int, Float)] = ShuffledRDD[9] at reduceByKey at <console>:31

scala> val minRevenuePerOrderId = orderItemsMap.reduceByKey((min,revenue) => if (min>revenue) revenue else min)

minRevenuePerOrderId: org.apache.spark.rdd.RDD[(Int, Float)] = ShuffledRDD[10] at reduceByKey at <console>:31

scala> minRevenuePerOrderId.take(10).foreach(println)

(65722,119.98)

(68522,329.99)

(23776,129.99)

(34207,99.96)

(53499,34.99)

(32676,59.99)

(53926,99.99)

(4926,199.92)

(38926,250.0)

(51620,99.96)

scala> minRevenuePerOrderId.sortByKey().take(10).foreach(println)

(1,299.98)

(2,129.99)

(4,49.98)

(5,99.96)

(7,79.95)

(8,50.0)

(9,199.98)

(10,21.99)

(11,49.98)

(12,100.0)

scala> revenuePerOrderId.sortByKey().take(10).foreach(println)

(1,299.98)

(2,579.98)

(4,699.85004)

(5,1129.8601)

(7,579.92004)

(8,729.84)

(9,599.96)

(10,651.92)

(11,919.79004)

(12,1299.8701)

scala>

//////////////////practice////

val orders=sc.textFile("public/retail_db/orders/part-00000")

val orderItems=sc.textFile("public/retail_db/order_items")

val ordersMap = orders.map(order => {(order.split(",")(0).toInt,order.split(",")(1).substring(0,10))

})

val orderItemsMap = orderItems.map(orderItem =>{

val oi = orderItem.split(",")

(oi(1).toInt,oi(4).toFloat)

})

//count of orders for each status

val orders=sc.textFile("public/retail_db/orders/part-00000")

//we can have key as 1 or null or 2 doesn't matter

orders.map(order => order.split("")(3)("1")).countByKey("").foreach(println)

```
//total revenue
val orderItems=sc.textFile("public/retail_db/order_items")
val orderItemsRevenue = orderItems.map(revenue => revenue.split(",")(4).toFloat)
orderItemsRevenue.reduce((total,revenue) => total+revenue)


//max revnue of an item
val orderItemsMaxRevenue = orderItemsRevenue.reduce((max,revenue) => {
if (max > revenue) max else revenue
})

//aggregation
val orderItems=sc.textFile("public/retail_db/order_items")
//get revenue per order_id
//get data in descending order by order_item_subtotal for each order_id

val orderItemsMap = orderItems.
map(oi => (oi.split(",")(1).toInt,oi.split(",")(4).toFloat))

val orderItemGBK = orderItemsMap.groupByKey
//Get revenue per order_id
orderItemGBK.map( rec => (rec._1, rec._2.toList.sum)).take(10).foreach(println)
```

```
// Get data in descending order by order_item_subtotal for each order_id


val ordersSortedbyRevenue =

orderItemGBK.

flatMap(rec => {

        rec._2.toList.sortBy(o => -o).map(k => (rec._1,k))

 })


ordersSortedbyRevenue.take(10).foreach(println)




val orderItems=sc.textFile("public/retail_db/order_items")

//get revenue per order_id

//get data in descending order by order_item_subtotal for each order_id


val orderItemsMap = orderItems.

map(oi => (oi.split(",")(1).toInt,oi.split(",")(4).toFloat))


val revenuePerOrderId = orderItemsMap.reduceByKey((total,revenue) => total+ revenue)


val minRevenuePerOrderId = orderItemsMap.reduceByKey((min,revenue) => if (min,revenue)
revenue else min)
```

```
//sorting

val products = sc.textFile("public/retail_db/products")

val productsMap = products.map( product => (product.split(",")(1).toInt,product))

val productssortedbycategoryid = productMap.sortByKey(false)


//sorting on 2 keys


//sorting

val products = sc.textFile("public/retail_db/products")

val productsMap = products.

filter(product => product.split(",")(4) !="").

map( product => ((product.split(",")(1).toInt,product.split(",")(4).toFloat),product))


val productssortedbycategoryid = productsMap.sortByKey().map(rec => rec._2)









//set operation
```

```
val orders=sc.textFile("public/retail_db/orders/part-00000")

val customer_201308 = orders.filter( order => order.split(",")(1).contains("2013-08")).map(order => order.split(",")(2).toInt)

val customer_201309 = orders.filter( order => order.split(",")(1).contains("2013-09")).map(order => order.split(",")(2).toInt)

val customer_uniq_2013= customer_201308.intersection(customer_201309)

val customer_union_2013= customer_201308.union(customer_201309)

 customer_union_2013.distinct.count

 //count by satus and save back to hdfs

//first method

 val orders=sc.textFile("public/retail_db/orders/part-00000")

 val order_count_by_status= orders.map(order => (order.split(",")(3),order.split(",")(2))).countByKey

 order_count_by_status.take(10).foreach(println)
```

```
sc.parallelize(order_count_by_status.toSeq).saveAsTextFile("/home/training/order_count_by_status1")


//second method

 val orders=sc.textFile("public/retail_db/orders/part-00000")

val  order_count_by_status= orders.map(order => (order.split(",")(3),1)).reduceByKey((total,element) => total+element )

order_count_by_status.map(rec => rec._1 + "\t" + rec._2).saveAsTextFile("/home/training/order_count_by_status2")



sc.textFile("/home/training/order_count_by_status/part-00000").collect.foreach(println)


//compression

val orders=sc.textFile("public/retail_db/orders/part-00000")

val  order_count_by_status= orders.map(order => (order.split(",")(3),1)).reduceByKey((total,element) => total+element )

order_count_by_status.saveAsTextFile("/home/training/order_count_by_status_snappy",classOf[org.apache.hadoop.io.compress.SnappyCodec] )



//practice
```

//Launch spark-shell

spark-shell --master yarn --num-executors 1 --executor-memory 512M --conf spark.ui.port=12673

//local mode

spark-shell

val orders=sc.textFile("public/retail_db/orders")

val orderItems=sc.textFile("public/retail_db/order_items")

orders.first

orders.take(10).foreach(println)

orderItems.first

orderItems.take(10).foreach(println)

//Filter for Completed or Closed orders

orders.map(order => order.split(",")(3)).distinct.collect.foreach(println) // check distinct status of orders

val ordersFilter = orders.filter(order=> order.split(",")(3)=="COMPLETE" || order.split(",")(3)=="CLOSED")

```
val ordersMap = ordersFilter.map( order => (order.split(",")(0).toInt,order.split(",")(1)))


val orderItemsMap = orderItems.
  map( oi => (oi.split(",")(1).toInt ,(oi.split(",")(2).toInt,oi.split(",")(4).toFloat)))


//join


val orderjoin = ordersMap.join(orderItemsMap)
```