Functional Programming

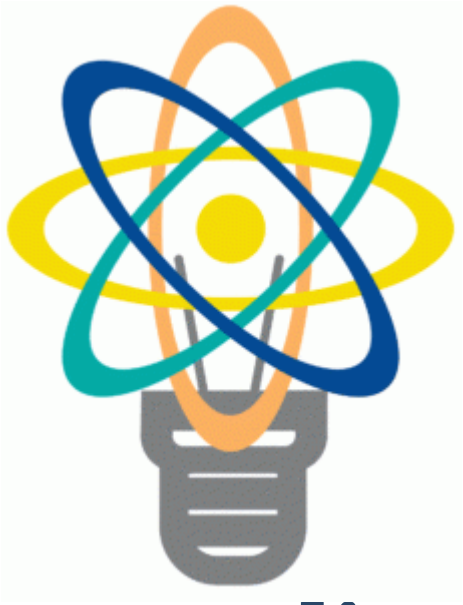
Luis Vásquez

About Me

- Big Data Advanced Developer
- Data Architect
- Oracle Sql Expert
- Oracle PL/SQL Profesional Developer
- MTA Introduction Python
  AWS Architect(In Progress)
- Python, C++, Java, PL/SQL, Power Builder, C#, Shell Script. – Cloudera Suite

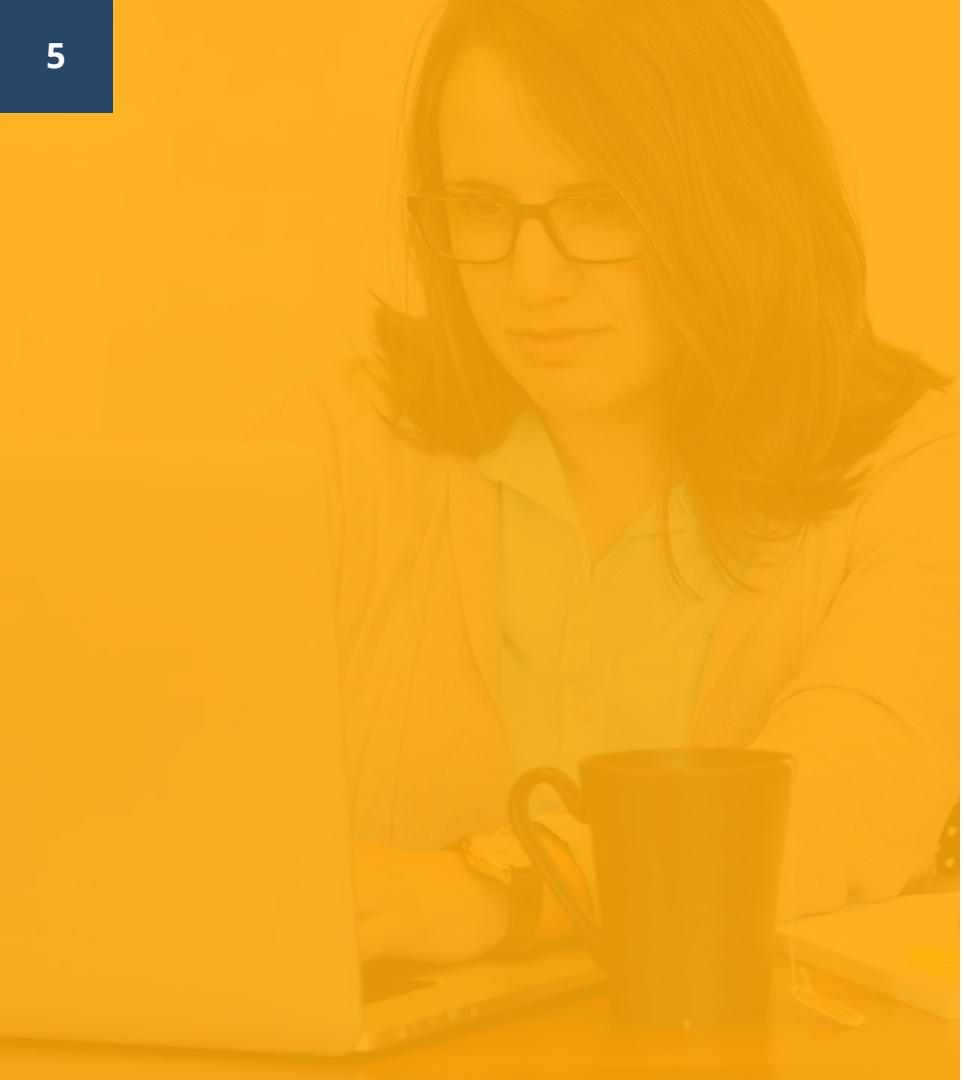
Topics


Paradigm
Functions
Introduction Functional Programming
Brief Introduction to Spark
Questions



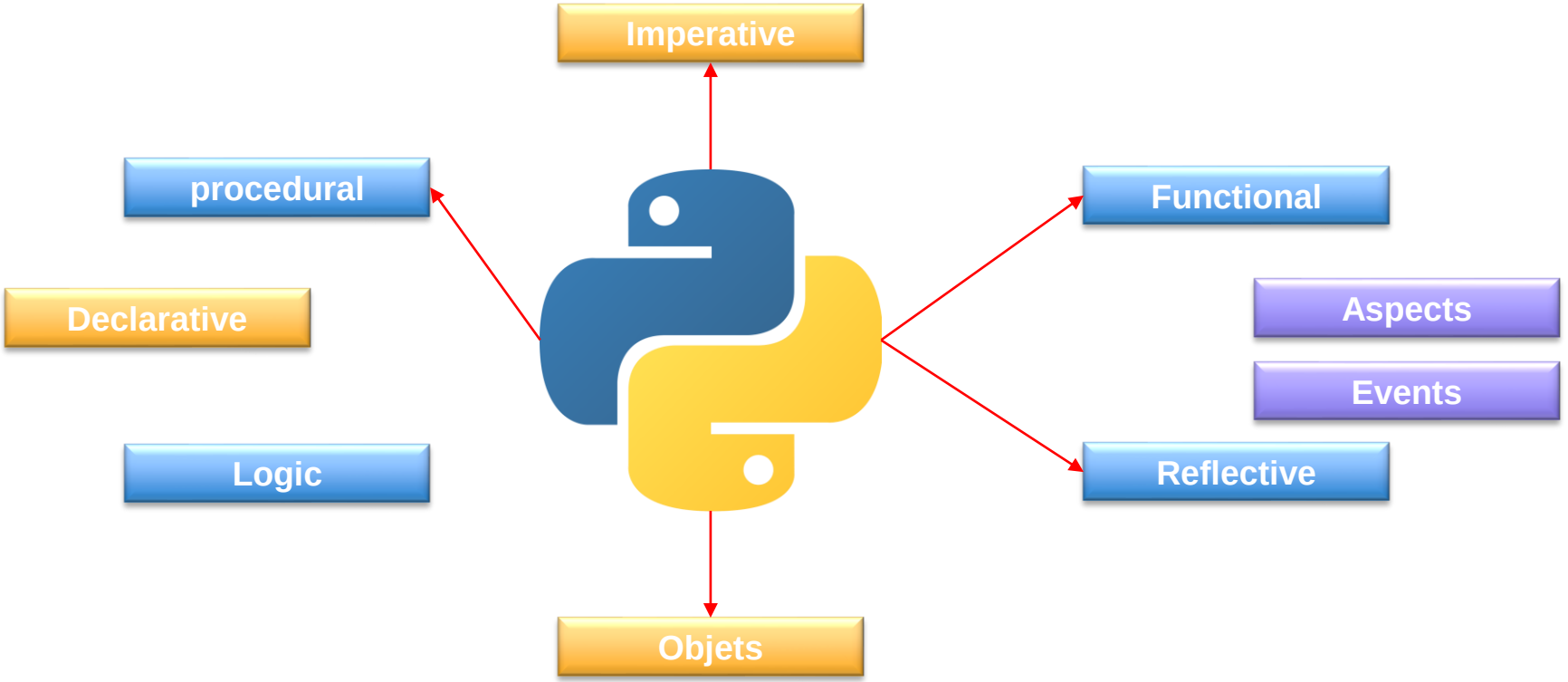
Paradigm

It’s a model to solve computational problems



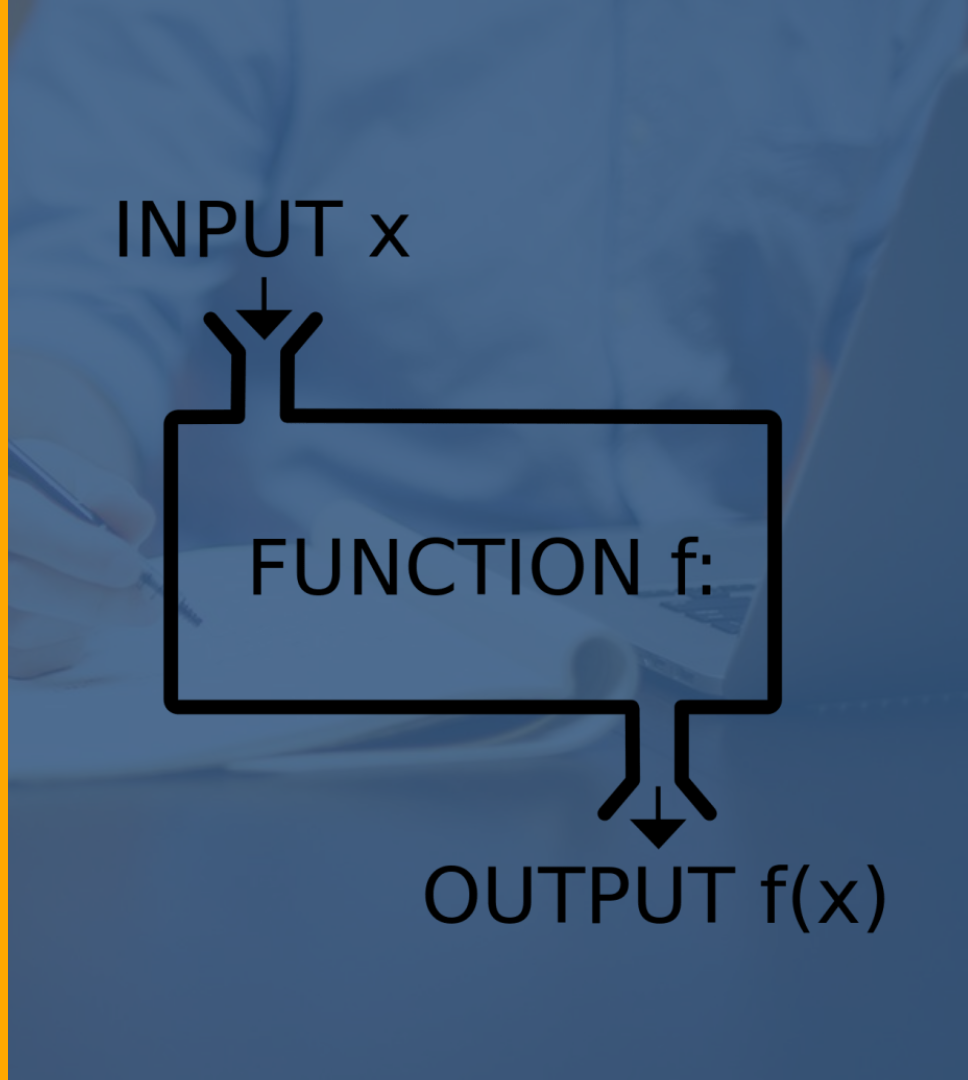
How many paradigms currently exist?

How many supports Python ?


Imperative
procedural
Declarative
Logic
Functional
Aspects
Events
Reflective
Objets

Example Functions Python

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data or function as a result



Lambda Function

Anonymous

```
1 print("The result is :",(lambda value1,value2 : value1 + value2)(1,3))
```

```
The result is : 4
```

Declarared

```
1 calc=lambda value1,value2 : value1 + value2
2 print(type(calc))
3 print("The result is :",calc(1,3))
```

```
<class 'function'>
The result is : 4
```

General Function Part 1

Void

```
1 def fn_sayhi():
2   print("hi! everyone")
3 fn_sayhi()
```

```
hi! everyone
```

Only Return

```
1 def fn_gethost():
2   return "192.168.1.72"
3 fn_gethost()
```

```
'192.168.1.72'
```

Parameters & Returns

```
1 def fn_getcontact(ar_rol,ar_status=0):
2   contact={'admin':'administrator@pythonmedellin.com',
3            'contact':'info@pythonmedellin.com'}
4   return contact[ar_rol],ar_status
5 fn_getcontact("admin")
```

```
('administrator@pythonmedellin.com', 0)
```

General Functions Part 2

Function as a parameter

```
1 def fn_sum(ar_value1,ar_value2):
2   return ar_value1 + ar_value2
3
4 def fn_calc(ar_function,ar_value1,ar_value2):
5   return ar_function(ar_value1,ar_value2)
6
7 print("declaraded : ",fn_calc(fn_sum,1,2))
8 print("anonym     : ",fn_calc(lambda value1,value2: value1 + value2,1,2))
```

General Function Part 3

Functions List

```
1 textdemo="PyThon Is Funy aNd Beautiful"
2 def fn_getinvert(ar_value):
3   return ar_value[::-1]
4
5 listfunctions=[str.lower,str.upper,fn_getinvert,lambda line: line.capitalize()]
6 def fn_apply(ar_list,ar_value):
7   for function in ar_list:
8     print("{0:13}: {1} :  ".format(function.__name__,function(ar_value)))
9 fn_apply(listfunctions,textdemo)
```

```
lower        : python is funy and beautiful :
upper        : PYTHON IS FUNY AND BEAUTIFUL :
fn_getinvert : lufituaeB dNa ynuF sI nohTyP :
<lambda>     : Python is funy and beautiful :
```

General Function Part 4

*Args

```
1 def fn_args(*ar_values):
2   accumulator=0
3   for value in ar_values:
4     accumulator+=value
5   return accumulator
6
7 fn_args(1,2,3,4,5,6,7,8)
```

```
36
```

**Kargs

```
1 def fn_kargs(**ar_values):
2   return ar_values.keys(),ar_values.values()
3 fn_kargs(language="python",conference="about functions")
4
```

```
(dict_keys(['language', 'conference']),
 dict_values(['python', 'about functions']))
```

General Functions Part 5

Wrapper

```
1 def fn_getfunction(ar_city):
2   function=None
3   def fn_medellin(ar_emoji):
4     return "python Medellín {}".format(ar_emoji)
5   def fn_bogota():
6     return "Python Bogotá".format(ar_emoji)
7
8   if ar_city == "05001":
9     function =fn_medellin
10  elif ar_city == "11001":
11    function =fn_bogota
12
13  return function
14
15 funccity=fn_getfunction("05001")
16 print("form # 1 :",funccity(":)"))
17 print("form # 2 :",fn_selectfunction("05001")(":)"))
```

```
form # 1 : python Medellín :)
form # 2 : python Medellín :)
```

General Functions Part 6

Closure

```
1 def fn_closure(ar_value):
2     def fn_acumulator(ar_acumulate):
3         return ar_acumulate + ar_value
4     return fn_acumulator
5
6 closure=fn_closure(5)
7
8 print("* first value closure  :",closure(85))
9 print("* second value closure :",closure(20))
10
```

```
* first value closure  : 90
* second value closure : 25
```

General Functions Part 7

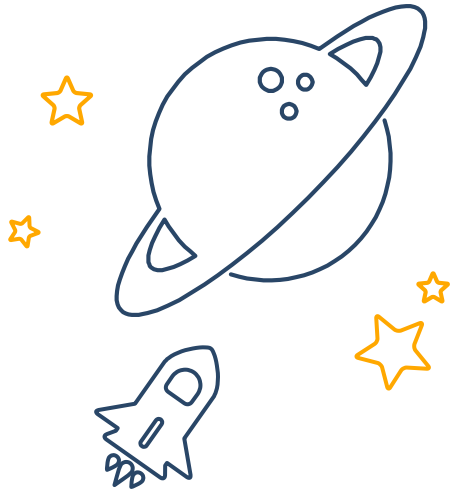
Decorators

```python
from datetime import datetime
def loguser(ar_function):
    def fn_wraper(*args,**kwargs):
        status="adult"
        print("*****************************************************")
        print("Welcome developers - Python Medellín")
        value=ar_function(*args,**kwargs)
        print("*****************************************************")
        if value < 18:
            status="younger"
        return status,ar_function(*args,**kwargs)
    return fn_wraper

@loguser
def fn_getage(ar_birthyear):
    return datetime.now().year - ar_birthyear

print("return value :",fn_getage(1987))
```

```
**************************************************
Welcome developers - Python Medellín
**************************************************
return value : ('adult', 32)
```



Introduction to Functional Programming

Functional Programming

```
1 values=["pass","yield","true","home","or","none"]
2 listletters=list(map(lambda value: value[0],values))
3 print("list :",listletters,", value :",''.join(listletters))
```

```
list : ['p', 'y', 't', 'h', 'o', 'n'] , value : python
```

Map

```
1 numbers=[1,2,3,4,5,6,7,8]
2 def fn_concat(ar_index):
3     return str(ar_index) +"sg!"
4
5 print("declared function  : ",list(map(fn_concat,numbers)))
6 print("anonymous function : ",list(map(lambda value:str(value) +"sg!",numbers)))
```

```
declared  :  ['1sg!', '2sg!', '3sg!', '4sg!', '5sg!', '6sg!', '7sg!', '8sg!']
anonymous :  ['1sg!', '2sg!', '3sg!', '4sg!', '5sg!', '6sg!', '7sg!', '8sg!']
```

Functional Programming

Filter

```
1 numbers=[1,2,3,4,5,6,7,8]
2 def fn_peers(ar_numer):
3     return  ar_numer % 2 == 0
4 print("result map         : ",list(map(fn_peers,numbers)))
5 print("declared function  : ",list(filter(fn_peers,numbers)))
6 print("anonymous function : ",list((filter(lambda number: number % 2 == 0,numbers))))
```

```
result map         :  [False, True, False, True, False, True, False, True]
declared function  :  [2, 4, 6, 8]
anonymous function :  [2, 4, 6, 8]
```

Functional Programming

Reduce

```
1 from functools import reduce
2 numbers=[9,2,3,8,5,6,7,8]
3 print("major number of list :",
4
5         reduce(lambda value,acum: value if value > acum else acum,numbers))
6 print("summation of list    :",
7           reduce(lambda value,acum:value +acum,numbers))
8
```

```
major number of list : 9
summation of list    : 48
```

Distributed Computing Example

Holidays in Colombia

holidays.csv

```
year;month;day;description_day;public_holiday
2019;1;1;MARTES;Y
2019;1;2;MIERCOLES;N
2019;1;3;JUEVEZ;N
2019;1;4;VIERNES;N
2019;1;5;SABADO;N
2019;1;6;DOMINGO;N
2019;1;7;LUNES;Y
2019;1;8;MARTES;N
2019;1;9;MIERCOLES;N
2019;1;10;JUEVEZ;N
```

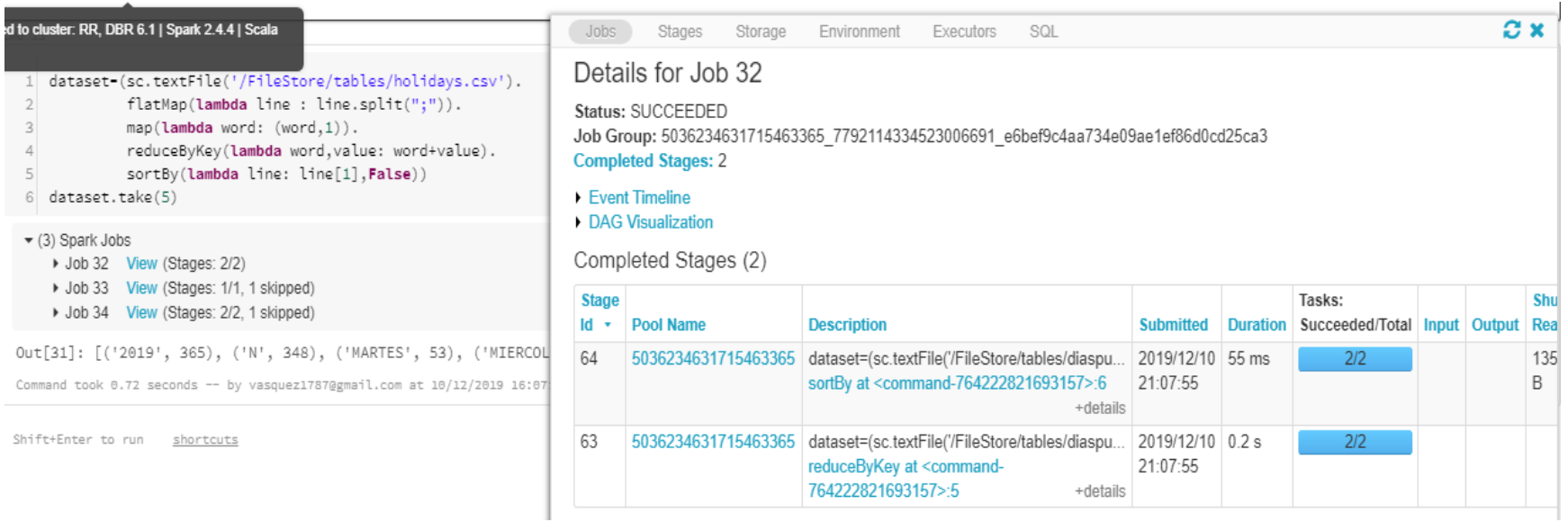
Holidays in Colombia

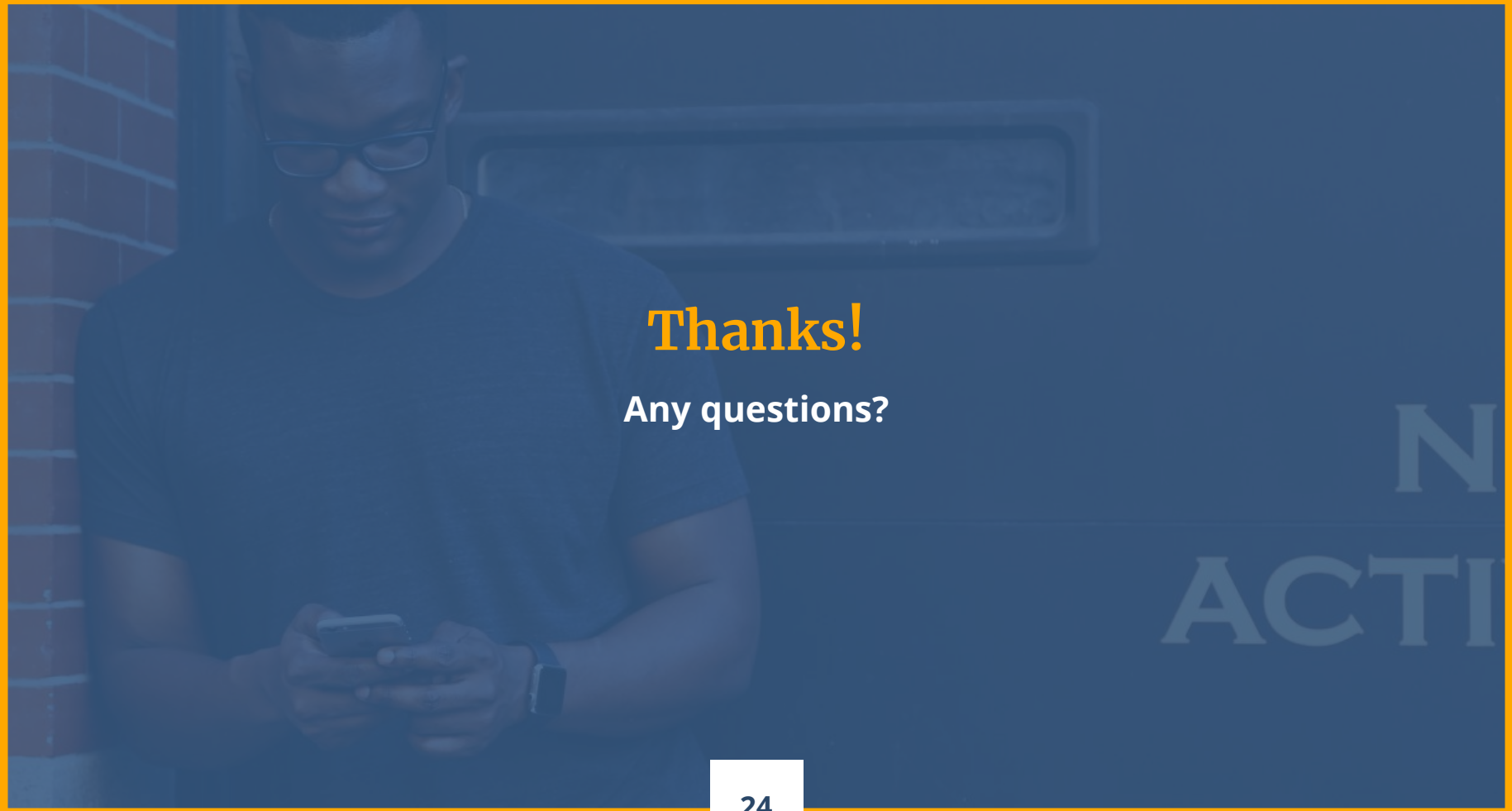
```
1  dataset=(sc.textFile('/FileStore/tables/holidays.csv').
2           flatMap(lambda line : line.split(";")).
3           map(lambda word: (word,1)).
4           reduceByKey(lambda word,value: word+value).
5           sortBy(lambda line: line[1],False))
6  dataset.take(5)
```

▸ (3) Spark Jobs

```
Out[31]: [('2019', 365), ('N', 348), ('MARTES', 53), ('MIERCOLES', 52), ('JUEVEZ', 52)]
```

Holidays in Colombia





Thanks!

Any questions?

Contact / Follow Me

@luisvasv @luisvasv luisvasv@gmail.com