OVERCOUNT()SUM()ROW\_NUMBER()RANK()DENSE\_RANK()LEAD()LAG()

DATE\_FORMAT(REGEXP\_REPLACE(DATA\_DT, ‘/’,’-’), ’yyyy-MM-dd’) data\_dt

Floor select distinct, insert into, update, delete, like ‘%1%’

Sort, Reduce内部排序，对reducer 随机not（real）

Distribute By partition (取余) 查询时

### Cluster By 查询时

当distribute by和sorts by字段相同时，可以使用cluster by方式。

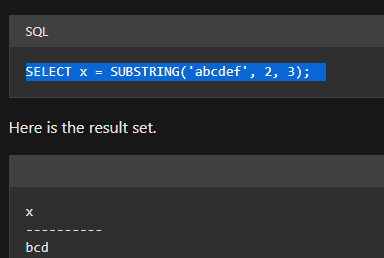
cluster by除了具有distribute by的功能外还兼具sort by的功能。但是排序只能是升序排序，不能指定排序规则为ASC或者DESC。

Partition by fe分区表 不是MR中的分区，存数据时候分的

怎么放数据？-HDFS 一个表映射（mapping）多个文件（分目录）

Debug看日志，go to log, check the exception

**String**CONCAT\_WS(separator, str1, str2,...)

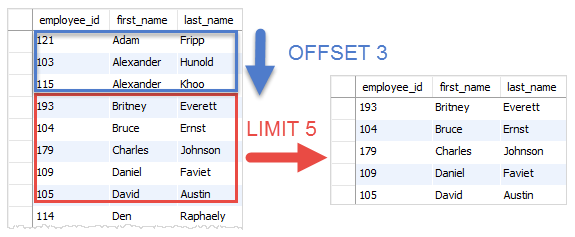


EXPLODE(col)：将hive一列中复杂的array或者map结构拆分成多行。

LATERAL VIEW

用法：LATERAL VIEW EXPLODE(split(,’,’)) tableAlias AS columnAlias

解释：用于和split, explode等UDTF一起使用，它能够将一列数据拆成多行数据，在此基础上可以对拆分后的数据进行聚合。



Over(): 加了单独的后:count(\*) sum over()， count sum查的是开窗后，over(空)为整表，

sum(cost) over(partition by name, month(orderdate)) 每人月购买总额

sum(cost) over(partition by name order by orderdate rows between UNBOUNDED PRECEDING and current row ) as sample4 ,--和sample3一样,由起点到当前行的聚合

或者 sum(cost) over(order by orderdate)

sum(cost) over(partition by name order by orderdate rows between 1 PRECEDING and current row) as sample5, --当前行和前面一行做聚合

sum(cost) over(partition by name order by orderdate rows between 1 PRECEDING AND 1 FOLLOWING ) as sample6,--当前行和前边一行及后面一行

sum(cost) over(partition by name order by orderdate rows between current row and UNBOUNDED FOLLOWING ) as sample7 --当前行及后面所有行

RANK() 排序相同时会重复，总数不会变

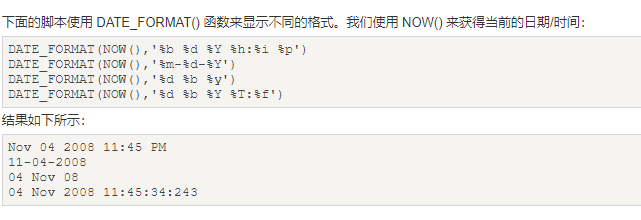
DENSE\_RANK() 排序相同时会重复，总数会减少

ROW\_NUMBER() 会根据顺序计算

lag(orderdate,2) over (partition by name order by orderdate) 上两条 lead

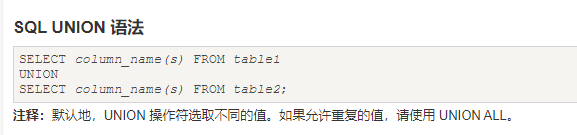
mysql> SET @one := 1

SELECT 1, 'reset' FROM DUAL WHERE (@found := NULL) IS NOT NULL;



DATEDIFF(datepart,startdate,enddate)

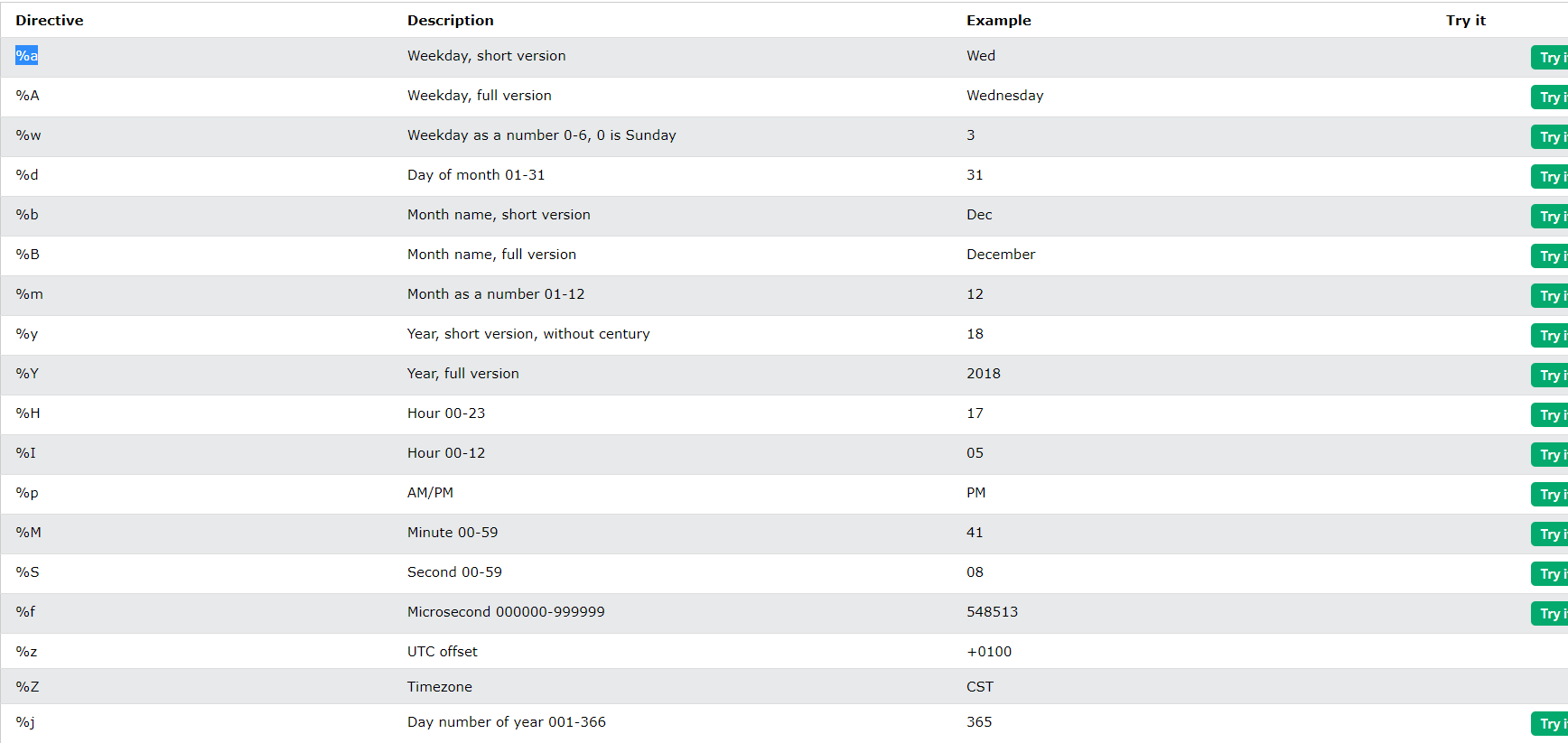
DATE(OrderDate)  提取日期

Python

x = datetime.datetime.now()  
print(x.year)

x = datetime.datetime(2020, 5, 17)  
print(x.strftime("%A"))



# Basic syntax:

from collections import defaultdict

your\_dict = defaultdict(data\_type)

# Where:

# - data\_type is the default empty data type that gets created if a

# key is passed to the dictionary that hasn't already been added

s.reverse()

Format: string.count(sub, start= 0,end=len(string))

string = "Add Grepper Answer"

print(string.count('e')

>>> 3

from collections import Counter

>>> Counter('abracadabra').most\_common(3)

[('a', 5), ('r', 2), ('b', 2)]

>>> heapq.heappush(heap, (3, 'create tests'))

>>> heapq.heappop(heap)#pops smallest

(1, 'write spec')

>>> heapq.nlargest(2,heap)#displays n largest values without popping

[(7, 'release product'),(5, 'write code')]

>>> heapq.nsmallest(2,heap)#displays n smallest values without popping

from collections import deque

students = deque(('Sarah', 'Mary', 'Carl'))

students.append('Charlotte') # this name appear at the end of the deque

students.appendleft('Ruth') # this name appear at the start of the deque

students.pop() # remove the element from the end

students.popleft() # remove the element from the start

print(students)

map(lambda x: x+2, [1,2,3,4]) #Output: [3,4,5,6]

仅仅一行即可搞定！

使用map和lambda迭代dictionary:

dict\_a = [{'name': 'python', 'points': 10}, {'name': 'java', 'points': 8}]

map(lambda x : x['name'], dict\_a) # Output: ['python', 'java']

map(lambda x : x['points']\*10, dict\_a) # Output: [100, 80]

map(lambda x : x['name'] == "python", dict\_a) # Output: [True, False]

以上代码中，dict\_a中的每个dict作为参数传递给lambda函数。lambda函数表达式作用于每个dict的结果作为输出。

map函数作用于多个iterables

list\_a = [1, 2, 3]

list\_b = [10, 20, 30]

map(lambda x, y: x + y, list\_a, list\_b) # Output: [11, 22, 33]

这里，list\_a和list\_b的第i个元素作为参数传递给lambda函数。

在Python3中，map函数返回一个惰性计算(lazily evaluated)的迭代器（iterator）或map对象。就像zip函数是惰性计算那样。  
我们不能通过index访问map对象的元素，也不能使用len()得到它的长度。  
但我们可以强制转换map对象为list：

map\_output = map(lambda x: x\*2, [1, 2, 3, 4])

print(map\_output) # Output: map object:

list\_map\_output = list(map\_output)

print(list\_map\_output) # Output: [2, 4, 6, 8]

## filter

filter的基本语法如下：

filter(function\_object, iterable)

filter函数需要两个参数，function\_object返回一个布尔值(boolean)，对iterable的每一个元素调用function\_object，filter只返回满足function\_object为True的元素。

和map函数一样，filter函数也返回一个list，但与map函数不同的是，filter函数只能有一个iterable作为输入。  
示例：  
返回偶数：

a = [1, 2, 3, 4, 5, 6]

filter(lambda x : x % 2 == 0, a) # Output: [2, 4, 6]

过滤dicts的list：

dict\_a = [{'name': 'python', 'points': 10}, {'name': 'java', 'points': 8}]

filter(lambda x : x['name'] == 'python', dict\_a)

# Output: [{'name': 'python', 'points': 10}]

和map一样，filter函数在Python3中返回一个惰性计算的filter对象或迭代器。我们不能通过index访问filter对象的元素，也不能使用len()得到它的长度。

list\_a = [1, 2, 3, 4, 5]

filter\_obj = filter(lambda x: x % 2 == 0, list\_a) # filter object

even\_num = list(filter\_obj) # Converts the filer obj to a list

print(even\_num) # Output: [2, 4]

def l**engthOfLongestSubstring**(self, s: str) -> int:

def check(start, end):

chars = [0] \* 128

for i in range(start, end + 1):

c = s[i]

chars[ord(c)] += 1

if chars[ord(c)] > 1:

return False

return True

n = len(s)

res = 0

for i in range(n):

for j in range(i, n):

if check(i, j):

res = max(res, j - i + 1)

return res

class Solution:

def **isValid**(self, s: str) -> bool:

stack =[]

mapping = {")":"(", "}":"{", "]":"["}

for char in s:

if char in mapping:

top\_char = stack.pop() if stack else '1'

if mapping[char] != top\_char:

return False

else:

stack.append(char)

return not stack

class Solution:

d**ef isAlienSorted**(self, words: List[str], order: str) -> bool:

order\_map = {}

for index, val in enumerate(order):

order\_map[val] = index

for i in range(len(words) - 1):

for j in range(len(words[i]) ):

if j > len(words[i + 1]) -1:

return False

if words[i][j] != words[i+1][j]:

if order\_map[words[i][j]] > order\_map[words[i+1][j]]:

return False

break

return True

class Solution:

def **countBinarySubstrings**(self, s: str) -> int:

prev,cur = 0,1

result = 0

for i in range(1,len(s)):

if s[i] == s[i-1]:

cur += 1

else:

prev, cur = cur, 1

if prev >= cur:

result += 1

return result

class Solution:

def **addBinary**(self, a: str, b: str) -> str:

result = '{0:b}'.format(int(a,2)+ int (b,2))

return result

def **firstUniqChar** (self, s: str) -> int:

count = collections.Counter(s)

for i, ch in enumerate(s):

if count[ch] == 1:

return i

return -1

class Solution:

def **isRobotBounded**(self, instructions: str) -> bool:

location = [0,0]

direction = ["E","S","W","N"]

pointer = 3

moves = {"E":[1,0],"S":[0,-1],"W":[-1,0],"N":[0,1]}

for i in instructions:

if i == "G":

location[0] = location[0] + moves[direction[pointer]][0]

location[1] = location[1] + moves[direction[pointer]][1]

elif i=="L":

pointer-=1

if pointer==-1:

pointer = 3

elif i== "R":

pointer +=1

if pointer == 4:

pointer = 0

#print(location)

return (location[0] == 0 and location[1] == 0) or pointer != 3

class Solution(object):

def **partitionLabels**(self, S):

last = {c: i for i, c in enumerate(S)}

j = anchor = 0

ans = []

for i, c in enumerate(S):

j = max(j, last[c])

if i == j:

ans.append(i - anchor + 1)

anchor = i + 1

return ans

class Solution:

def topKFrequent(self, words: List[str], k: int) -> List[str]:

wordCount = dict()

for w in words:

if w in wordCount:

wordCount[w] += 1

else:

wordCount[w] = 1

h = []

heapify(h)

for key, v in wordCount.items():

heappush(h, (-v, key))

res = []

for i in range(k):

res.append(heappop(h)[1])

return res