Left-anti join with null value

NULL value exception in pyspark and how to handle

認知 NULL值在spark是特殊的

example 1

In [25]:
```
df=spark.createDataFrame([('a',1,1),('b',2,1),('c',1,None)],
                         schema=('id', 'foo', 'bar')
                        )
df.show()
```

```
StatementMeta(sparkcluster01, 10963, 26, Finished, Available)
+---+---+----+
| id|foo| bar|
+---+---+----+
|  a|  1|   1|
|  b|  2|   1|
|  c|  1|null|
+---+---+----+
```

Target: 想要找出foo=1 且 bar不等於1的id，預期要是id=c

In [16]:
```
#測試
df.filter( (df.foo==1) & (df.bar!=1) ).show()
```

```
StatementMeta(sparkcluster01, 10963, 17, Finished, Available)
+---+---+---+
| id|foo|bar|
+---+---+---+
+---+---+---+
```

預期: id=c bar值為null不等於1，且foo=1，應該要取得id=c的資料

實際: 沒有配對到任何資料

結果: 失敗

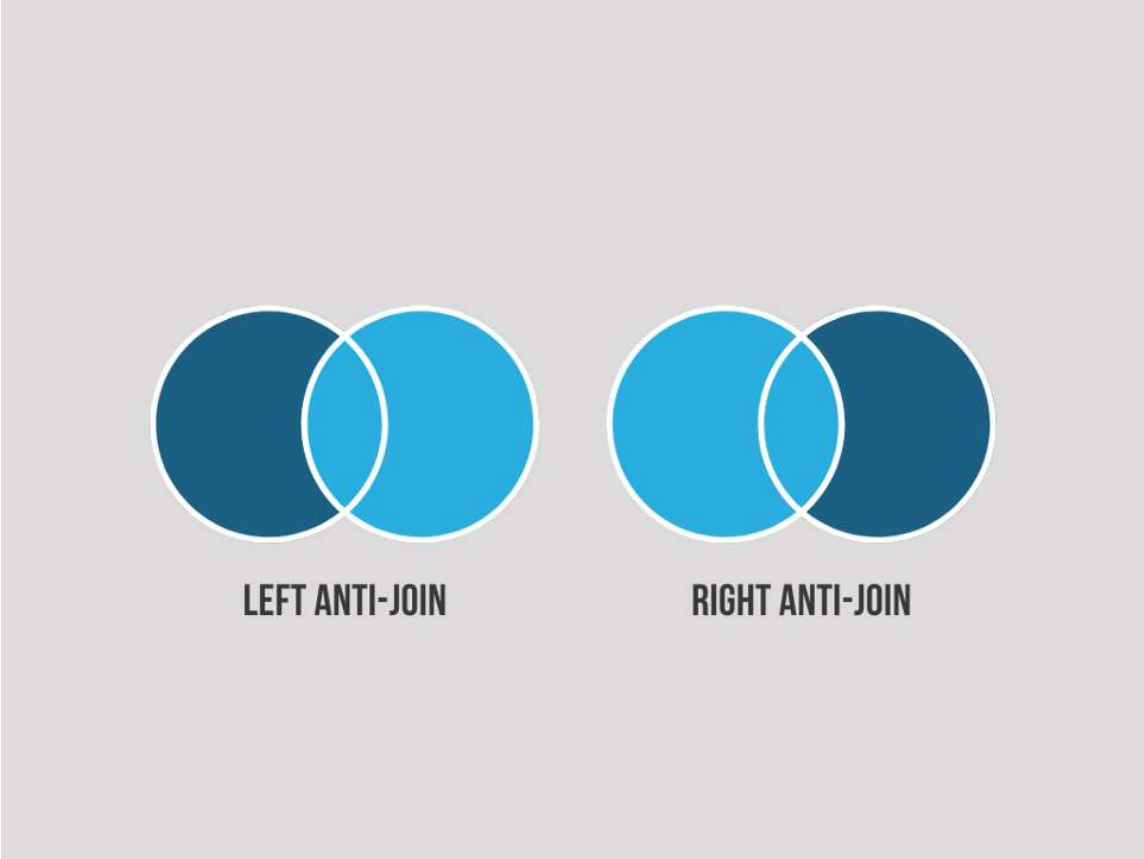
example 2

In [27]:
```
df1=spark.createDataFrame([('test1',19,5),('test2',0,19),('test3',None,95)],
                         schema=('id1','id2','value')
                        )

df2=spark.createDataFrame([('test3',None,5),('test22',5,9487)],
                         schema=('id1','id2','value')
                        )

df1.show()
df2.show()
```

```
StatementMeta(sparkcluster01, 10963, 28, Finished, Available)
```

```
+-----+----+-----+
|  id1| id2|value|
+-----+----+-----+
|test1|  19|    5|
|test2|   0|   19|
|test3|null|   95|
+-----+----+-----+

+------+----+-----+
|   id1| id2|value|
+------+----+-----+
| test3|null|    5|
|test22|   5| 9487|
+------+----+-----+
```

希望用delete insert方式結合B資料，使用join的leftanti方式，在union B



In [30]:
```
df1.join(df2,['id1','id2'],how='leftanti').union(df2).show()
```

```
StatementMeta(sparkcluster01, 10963, 31, Finished, Available)
+------+----+-----+
|   id1| id2|value|
+------+----+-----+
| test3|null|   95|
| test2|   0|   19|
| test1|  19|    5|
| test3|null|    5|
|test22|   5| 9487|
+------+----+-----+
```

預期: df1與df2共同擁有id1=test3, id2=Null，為重複key值，在A.join(left-anti)時會捨棄df1的id=test3，union df2補回來id1=test3， 更新id1=test3, id2=None的 value。

實際: df1與df2的id1=test3,id2=None 兩筆資料都保留下來

結果: 失敗

小結: NULL 沒辦法判斷大小、等於非等於，只能使用isNull isNotNull來判斷是否是NULL值，其餘為未定義

官方解方 <=> or eqNullSafe

參考資料:

NULL semantic

eqNullSafe

Example 1 : 正確語法

In [31]:
```
df.filter( (df.foo==1) & ((df.bar.isNull()) | (df.bar!=1)) ).show()
```

```
StatementMeta(sparkcluster01, 10963, 32, Finished, Available)
+---+---+----+
| id|foo| bar|
+---+---+----+
|  c|  1|null|
+---+---+----+
```

In [32]:
```
df.filter( (df.foo==1) & ~(df.bar.eqNullSafe(1)) ).show()
```

```
StatementMeta(sparkcluster01, 10963, 33, Finished, Available)
+---+---+----+
| id|foo| bar|
+---+---+----+
|  c|  1|null|
+---+---+----+
```

Example 2: 正確語法

In [33]:
```
_key_col= ['id1','id2']

join_cond= [df1[k].eqNullSafe(df2[k]) for k in _key_col] #讓NULL可以判斷等於NULL 透過

join_cond
```

```
StatementMeta(sparkcluster01, 10963, 34, Finished, Available)
```

Out[33]:
```
[Column<'(id1 <=> id1)'>, Column<'(id2 <=> id2)'>]
```

In [34]:
```
df1.join(df2,join_cond,how='leftanti').union(df2).show()
```

```
StatementMeta(sparkcluster01, 10963, 35, Finished, Available)
```

```
+------+----+-----+
|   id1| id2|value|
+------+----+-----+
| test2|   0|   19|
| test1|  19|    5|
| test3|null|    5|
|test22|   5| 9487|
+------+----+-----+
```

小結: 透過上面可以找出如何使用官方提供方式，讓**NULL = NULL** 是**True**，而非未定義，進而得到預期結果

結論:

當判斷的col有NULL值時，任何邏輯判斷需要特別特別注意，很容易出現異想不倒的情況，需要提前排除，透過官方提供方式，能夠有效避免問題