In [1]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-25-ML-preprocessing")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [2]:
```
sales = spark.read.format("csv")\
  .option("header", "true")\
  .option("inferSchema", "true")\
  .load(SPARK_BOOK_DATA_PATH + "/data/retail-data/by-day/*.csv")\
  .coalesce(5)\
  .where("Description IS NOT NULL")
```

In [3]:
```
sales.show(3, False)
```

```
+---------+---------+-----------------------------+--------+-------------------+---------+-------
---+--------------+
|InvoiceNo|StockCode|Description                  |Quantity|InvoiceDate        |UnitPrice|Custome
rID|Country       |
+---------+---------+-----------------------------+--------+-------------------+---------+-------
---+--------------+
|580538   |23084    |RABBIT NIGHT LIGHT           |48      |2011-12-05 08:38:00|1.79     |14075.0
|United Kingdom|
|580538   |23077    |DOUGHNUT LIP GLOSS           |20      |2011-12-05 08:38:00|1.25     |14075.0
|United Kingdom|
|580538   |22906    |12 MESSAGE CARDS WITH ENVELOPES|24      |2011-12-05 08:38:00|1.65     |14075.0
|United Kingdom|
+---------+---------+-----------------------------+--------+-------------------+---------+-------
---+--------------+
only showing top 3 rows
```

In [4]:
```
fakeIntDF = spark.read.parquet(SPARK_BOOK_DATA_PATH + "/data/simple-ml-integers")
```

In [6]:
```
fakeIntDF.show(5, False)
```

```
+----+----+----+
|int1|int2|int3|
+----+----+----+
|1   |2   |3   |
|7   |8   |9   |
|4   |5   |6   |
+----+----+----+
```

In [7]:
```
simpleDF = spark.read.json(SPARK_BOOK_DATA_PATH + "/data/simple-ml")
```

In [8]:
```
simpleDF.show(5, False)
```

```
+-----+----+------+------------------+
|color|lab |value1|value2            |
+-----+----+------+------------------+
|green|good|1     |14.386294994851129|
|blue |bad |8     |14.386294994851129|
|blue |bad |12    |14.386294994851129|
|green|good|15    |38.97187133755819 |
|green|good|12    |14.386294994851129|
+-----+----+------+------------------+
only showing top 5 rows
```

In [9]:
```
scaleDF = spark.read.parquet(SPARK_BOOK_DATA_PATH + "/data/simple-ml-scaling")
```

In [10]:
```
scaleDF.show(5, False)
```

```
+---+--------------+
|id |features      |
+---+--------------+
|0  |[1.0,0.1,-1.0]|
|1  |[2.0,1.1,1.0] |
|0  |[1.0,0.1,-1.0]|
|1  |[2.0,1.1,1.0] |
|1  |[3.0,10.1,3.0]|
+---+--------------+
```

In [11]:
```
# COMMAND ----------

from pyspark.ml.feature import RFormula

supervised = RFormula(formula="lab ~ . + color:value1 + color:value2")
supervised.fit(simpleDF).transform(simpleDF).show(5, False)
```

```
+-----+----+------+------------------+-----------------------------------------------------------
---------+-----+
|color|lab |value1|value2            |features
|label|
+-----+----+------+------------------+-----------------------------------------------------------
---------+-----+
|green|good|1     |14.386294994851129|(10,[1,2,3,5,8],[1.0,1.0,14.386294994851129,1.0,14.3862949948
51129])  |1.0  |
|blue |bad |8     |14.386294994851129|(10,[2,3,6,9],[8.0,14.386294994851129,8.0,14.38629499485112
9])       |0.0  |
|blue |bad |12    |14.386294994851129|(10,[2,3,6,9],[12.0,14.386294994851129,12.0,14.38629499485112
9])      |0.0  |
|green|good|15    |38.97187133755819 |(10,[1,2,3,5,8],[1.0,15.0,38.97187133755819,15.0,38.971871337
55819])  |1.0  |
|green|good|12    |14.386294994851129|(10,[1,2,3,5,8],[1.0,12.0,14.386294994851129,12.0,14.38629499
4851129])|1.0  |
+-----+----+------+------------------+-----------------------------------------------------------
---------+-----+
only showing top 5 rows
```

In [12]:
```
# COMMAND ----------

from pyspark.ml.feature import SQLTransformer

basicTransformation = SQLTransformer()\
  .setStatement("""
    SELECT sum(Quantity), count(*), CustomerID
    FROM __THIS__
    GROUP BY CustomerID
  """)

basicTransformation.transform(sales).show(5, False)
```

```
+-------------+--------+----------+
|sum(Quantity)|count(1)|CustomerID|
+-------------+--------+----------+
|119          |62      |14452.0   |
|440          |143     |16916.0   |
|630          |72      |17633.0   |
|34           |6       |14768.0   |
|1542         |30      |13094.0   |
+-------------+--------+----------+
only showing top 5 rows
```

In [13]:
```
# COMMAND ----------

from pyspark.ml.feature import VectorAssembler
va = VectorAssembler().setInputCols(["int1", "int2", "int3"])
va.transform(fakeIntDF).show(5, False)
```

```
+----+----+----+-----------------------------------+
|int1|int2|int3|VectorAssembler_e89e9cfd952e__output|
+----+----+----+-----------------------------------+
|1   |2   |3   |[1.0,2.0,3.0]                      |
|7   |8   |9   |[7.0,8.0,9.0]                      |
|4   |5   |6   |[4.0,5.0,6.0]                      |
+----+----+----+-----------------------------------+
```

In [14]:
```
# COMMAND ----------

contDF = spark.range(20).selectExpr("cast(id as double)")
contDF.show(5, False)
```

```
+---+
|id |
+---+
|0.0|
|1.0|
|2.0|
|3.0|
|4.0|
+---+
only showing top 5 rows
```

In [18]:
```
contDF.show?
```

In [19]:
```
# COMMAND ----------

from pyspark.ml.feature import Bucketizer
bucketBorders = [-1.0, 5.0, 10.0, 250.0, 600.0]
bucketer = Bucketizer().setSplits(bucketBorders).setInputCol("id")
bucketer.transform(contDF).show(truncate=False)
```

```
+----+--------------------------------+
|id  |Bucketizer_67efb63d26d8__output|
+----+--------------------------------+
|0.0 |0.0                             |
|1.0 |0.0                             |
|2.0 |0.0                             |
|3.0 |0.0                             |
|4.0 |0.0                             |
|5.0 |1.0                             |
|6.0 |1.0                             |
|7.0 |1.0                             |
|8.0 |1.0                             |
|9.0 |1.0                             |
|10.0|2.0                             |
|11.0|2.0                             |
|12.0|2.0                             |
|13.0|2.0                             |
|14.0|2.0                             |
|15.0|2.0                             |
|16.0|2.0                             |
|17.0|2.0                             |
|18.0|2.0                             |
|19.0|2.0                             |
+----+--------------------------------+
```

In [21]:
```
# COMMAND ----------

from pyspark.ml.feature import QuantileDiscretizer
bucketer = QuantileDiscretizer().setInputCol("id").setNumBuckets(5)
```

In [25]:
```
## not working
## bucketer.fit(contDF).transform(contDF).show(truncate=False)
```

In [26]:
```
# COMMAND ----------

from pyspark.ml.feature import StandardScaler
sScaler = StandardScaler().setInputCol("features")
sScaler.fit(scaleDF).transform(scaleDF).show(5, False)
```

```
+---+--------------+------------------------------------------------------------+
|id |features      |StandardScaler_26946cf17d37__output                         |
+---+--------------+------------------------------------------------------------+
|0  |[1.0,0.1,-1.0]|[1.1952286093343936,0.02337622911060922,-0.5976143046671968]|
|1  |[2.0,1.1,1.0] |[2.390457218668787,0.2571385202167014,0.5976143046671968]   |
|0  |[1.0,0.1,-1.0]|[1.1952286093343936,0.02337622911060922,-0.5976143046671968]|
|1  |[2.0,1.1,1.0] |[2.390457218668787,0.2571385202167014,0.5976143046671968]   |
|1  |[3.0,10.1,3.0]|[3.5856858280031805,2.3609991401715313,1.7928429140015902]  |
+---+--------------+------------------------------------------------------------+
```

In [27]:
```
# COMMAND ----------

from pyspark.ml.feature import MinMaxScaler
minMax = MinMaxScaler().setMin(5).setMax(10).setInputCol("features")
minMax.fit(scaleDF).transform(scaleDF).show(5, False)
```

```
+---+--------------+---------------------------------+
|id |features      |MinMaxScaler_7eb3957ba1c1__output|
+---+--------------+---------------------------------+
|0  |[1.0,0.1,-1.0]|[5.0,5.0,5.0]                    |
|1  |[2.0,1.1,1.0] |[7.5,5.5,7.5]                    |
|0  |[1.0,0.1,-1.0]|[5.0,5.0,5.0]                    |
|1  |[2.0,1.1,1.0] |[7.5,5.5,7.5]                    |
|1  |[3.0,10.1,3.0]|[10.0,10.0,10.0]                 |
+---+--------------+---------------------------------+
```

In [28]:
```
# COMMAND ----------

from pyspark.ml.feature import MaxAbsScaler
maScaler = MaxAbsScaler().setInputCol("features")
maScaler.fit(scaleDF).transform(scaleDF).show(5, False)
```

```
+---+--------------+------------------------------------------------------------+
|id |features      |MaxAbsScaler_0ca09d05628f__output                           |
+---+--------------+------------------------------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.3333333333333333,0.009900990099009901,-0.3333333333333333]|
|1  |[2.0,1.1,1.0] |[0.6666666666666666,0.10891089108910892,0.3333333333333333]  |
|0  |[1.0,0.1,-1.0]|[0.3333333333333333,0.009900990099009901,-0.3333333333333333]|
|1  |[2.0,1.1,1.0] |[0.6666666666666666,0.10891089108910892,0.3333333333333333]  |
|1  |[3.0,10.1,3.0]|[1.0,1.0,1.0]                                               |
+---+--------------+------------------------------------------------------------+
```

In [29]:
```
# COMMAND ----------

from pyspark.ml.feature import ElementwiseProduct
from pyspark.ml.linalg import Vectors
scaleUpVec = Vectors.dense(10.0, 15.0, 20.0)
scalingUp = ElementwiseProduct()\
  .setScalingVec(scaleUpVec)\
  .setInputCol("features")
scalingUp.transform(scaleDF).show(5, False)
```

```
+---+--------------+---------------------------------------+
|id |features      |ElementwiseProduct_b453d056065a__output|
+---+--------------+---------------------------------------+
|0  |[1.0,0.1,-1.0]|[10.0,1.5,-20.0]                       |
|1  |[2.0,1.1,1.0] |[20.0,16.5,20.0]                       |
|0  |[1.0,0.1,-1.0]|[10.0,1.5,-20.0]                       |
|1  |[2.0,1.1,1.0] |[20.0,16.5,20.0]                       |
|1  |[3.0,10.1,3.0]|[30.0,151.5,60.0]                      |
+---+--------------+---------------------------------------+
```

In [30]:
```
# COMMAND ----------

from pyspark.ml.feature import Normalizer
manhattanDistance = Normalizer().setP(1).setInputCol("features")
manhattanDistance.transform(scaleDF).show(5, False)
```

```
+---+-------------+-----------------------------------------------------------+
|id |features     |Normalizer_fb01d7f40623__output                            |
+---+-------------+-----------------------------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.47619047619047616,0.047619047619047616,-0.47619047619047616]|
|1  |[2.0,1.1,1.0] |[0.48780487804878053,0.26829268292682934,0.24390243902439027]  |
|0  |[1.0,0.1,-1.0]|[0.47619047619047616,0.047619047619047616,-0.47619047619047616]|
|1  |[2.0,1.1,1.0] |[0.48780487804878053,0.26829268292682934,0.24390243902439027]  |
|1  |[3.0,10.1,3.0]|[0.18633540372670807,0.6273291925465838,0.18633540372670807]   |
+---+-------------+-----------------------------------------------------------+
```

In [31]:
```
# COMMAND ----------

from pyspark.ml.feature import StringIndexer
lblIndxr = StringIndexer().setInputCol("lab").setOutputCol("labelInd")
idxRes = lblIndxr.fit(simpleDF).transform(simpleDF)
idxRes.show(5, False)
```

```
+-----+----+------+------------------+--------+
|color|lab |value1|value2            |labelInd|
+-----+----+------+------------------+--------+
|green|good|1     |14.386294994851129|1.0     |
|blue |bad |8     |14.386294994851129|0.0     |
|blue |bad |12    |14.386294994851129|0.0     |
|green|good|15    |38.97187133755819 |1.0     |
|green|good|12    |14.386294994851129|1.0     |
+-----+----+------+------------------+--------+
only showing top 5 rows
```

In [33]:
```
# COMMAND ----------

from pyspark.ml.feature import IndexToString
labelReverse = IndexToString().setInputCol("labelInd")
labelReverse.transform(idxRes).show(5, False)
```

```
+-----+----+------+------------------+--------+-----------------------------------+
|color|lab |value1|value2            |labelInd|IndexToString_4aaec40703f8__output|
+-----+----+------+------------------+--------+-----------------------------------+
|green|good|1     |14.386294994851129|1.0     |good                               |
|blue |bad |8     |14.386294994851129|0.0     |bad                                |
|blue |bad |12    |14.386294994851129|0.0     |bad                                |
|green|good|15    |38.97187133755819 |1.0     |good                               |
|green|good|12    |14.386294994851129|1.0     |good                               |
+-----+----+------+------------------+--------+-----------------------------------+
only showing top 5 rows
```

In [34]:
```
# COMMAND ----------

valIndexer = StringIndexer().setInputCol("value1").setOutputCol("value1_Ind")
valIndexer.fit(simpleDF).transform(simpleDF).show(5, False)
```

```
+-----+----+------+------------------+----------+
|color|lab |value1|value2            |value1_Ind|
+-----+----+------+------------------+----------+
|green|good|1     |14.386294994851129|2.0       |
|blue |bad |8     |14.386294994851129|4.0       |
|blue |bad |12    |14.386294994851129|0.0       |
|green|good|15    |38.97187133755819 |5.0       |
|green|good|12    |14.386294994851129|0.0       |
+-----+----+------+------------------+----------+
only showing top 5 rows
```

In [35]:
```
# COMMAND ----------

from pyspark.ml.feature import VectorIndexer
from pyspark.ml.linalg import Vectors
idxIn = spark.createDataFrame([
  (Vectors.dense(1, 2, 3),1),
  (Vectors.dense(2, 5, 6),2),
  (Vectors.dense(1, 8, 9),3)
]).toDF("features", "label")
indxr = VectorIndexer()\
  .setInputCol("features")\
  .setOutputCol("idxed")\
  .setMaxCategories(2)
indxr.fit(idxIn).transform(idxIn).show(5, False)
```

```
+-------------+-----+-------------+
|features     |label|idxed        |
+-------------+-----+-------------+
|[1.0,2.0,3.0]|1    |[0.0,2.0,3.0]|
|[2.0,5.0,6.0]|2    |[1.0,5.0,6.0]|
|[1.0,8.0,9.0]|3    |[0.0,8.0,9.0]|
+-------------+-----+-------------+
```

In [36]:
```
# COMMAND ----------

from pyspark.ml.feature import OneHotEncoder, StringIndexer
lblIndxr = StringIndexer().setInputCol("color").setOutputCol("colorInd")
colorLab = lblIndxr.fit(simpleDF).transform(simpleDF.select("color"))
ohe = OneHotEncoder().setInputCol("colorInd")
ohe.transform(colorLab).show(5, False)
```

```
+-----+--------+---------------------------------+
|color|colorInd|OneHotEncoder_77a2e2f63a02__output|
+-----+--------+---------------------------------+
|green|1.0     |(2,[1],[1.0])                    |
|blue |2.0     |(2,[],[])                        |
|blue |2.0     |(2,[],[])                        |
|green|1.0     |(2,[1],[1.0])                    |
|green|1.0     |(2,[1],[1.0])                    |
+-----+--------+---------------------------------+
only showing top 5 rows
```

In [37]:
```
# COMMAND ----------

from pyspark.ml.feature import Tokenizer
tkn = Tokenizer().setInputCol("Description").setOutputCol("DescOut")
tokenized = tkn.transform(sales.select("Description"))
tokenized.show(20, False)
```

```
+---------------------------------+-----------------------------------------+
|Description                      |DescOut                                  |
+---------------------------------+-----------------------------------------+
|RABBIT NIGHT LIGHT               |[rabbit, night, light]                   |
|DOUGHNUT LIP GLOSS               |[doughnut, lip, gloss]                   |
|12 MESSAGE CARDS WITH ENVELOPES  |[12, message, cards, with, envelopes]    |
|BLUE HARMONICA IN BOX            |[blue, harmonica, in, box]               |
|GUMBALL COAT RACK                |[gumball, coat, rack]                    |
|SKULLS  WATER TRANSFER TATTOOS   |[skulls, , water, transfer, tattoos]     |
|FELTCRAFT GIRL AMELIE KIT        |[feltcraft, girl, amelie, kit]           |
|CAMOUFLAGE LED TORCH             |[camouflage, led, torch]                 |
|WHITE SKULL HOT WATER BOTTLE     |[white, skull, hot, water, bottle]       |
|ENGLISH ROSE HOT WATER BOTTLE    |[english, rose, hot, water, bottle]      |
|HOT WATER BOTTLE KEEP CALM       |[hot, water, bottle, keep, calm]         |
|SCOTTIE DOG HOT WATER BOTTLE     |[scottie, dog, hot, water, bottle]       |
|ROSE CARAVAN DOORSTOP            |[rose, caravan, doorstop]                |
|GINGHAM HEART  DOORSTOP RED      |[gingham, heart, , doorstop, red]        |
|STORAGE TIN VINTAGE LEAF         |[storage, tin, vintage, leaf]            |
|SET OF 4 KNICK KNACK TINS POPPIES|[set, of, 4, knick, knack, tins, poppies]|
|POPCORN HOLDER                   |[popcorn, holder]                        |
|GROW A FLYTRAP OR SUNFLOWER IN TIN|[grow, a, flytrap, or, sunflower, in, tin]|
|AIRLINE BAG VINTAGE WORLD CHAMPION|[airline, bag, vintage, world, champion]|
|AIRLINE BAG VINTAGE JET SET BROWN|[airline, bag, vintage, jet, set, brown]|
+---------------------------------+-----------------------------------------+
only showing top 20 rows
```

In [38]:
```
# COMMAND ----------

from pyspark.ml.feature import RegexTokenizer
rt = RegexTokenizer()\
  .setInputCol("Description")\
  .setOutputCol("DescOut")\
  .setPattern(" ")\
  .setToLowercase(True)
rt.transform(sales.select("Description")).show(20, False)
```

```
+-----------------------------------+-----------------------------------------+
|Description                        |DescOut                                  |
+-----------------------------------+-----------------------------------------+
|RABBIT NIGHT LIGHT                 |[rabbit, night, light]                   |
|DOUGHNUT LIP GLOSS                 |[doughnut, lip, gloss]                   |
|12 MESSAGE CARDS WITH ENVELOPES    |[12, message, cards, with, envelopes]    |
|BLUE HARMONICA IN BOX              |[blue, harmonica, in, box]               |
|GUMBALL COAT RACK                  |[gumball, coat, rack]                    |
|SKULLS  WATER TRANSFER TATTOOS     |[skulls, water, transfer, tattoos]       |
|FELTCRAFT GIRL AMELIE KIT          |[feltcraft, girl, amelie, kit]           |
|CAMOUFLAGE LED TORCH               |[camouflage, led, torch]                 |
|WHITE SKULL HOT WATER BOTTLE       |[white, skull, hot, water, bottle]       |
|ENGLISH ROSE HOT WATER BOTTLE      |[english, rose, hot, water, bottle]      |
|HOT WATER BOTTLE KEEP CALM         |[hot, water, bottle, keep, calm]         |
|SCOTTIE DOG HOT WATER BOTTLE       |[scottie, dog, hot, water, bottle]       |
|ROSE CARAVAN DOORSTOP              |[rose, caravan, doorstop]                |
|GINGHAM HEART  DOORSTOP RED        |[gingham, heart, doorstop, red]          |
|STORAGE TIN VINTAGE LEAF           |[storage, tin, vintage, leaf]            |
|SET OF 4 KNICK KNACK TINS POPPIES  |[set, of, 4, knick, knack, tins, poppies]|
|POPCORN HOLDER                     |[popcorn, holder]                        |
|GROW A FLYTRAP OR SUNFLOWER IN TIN |[grow, a, flytrap, or, sunflower, in, tin]|
|AIRLINE BAG VINTAGE WORLD CHAMPION |[airline, bag, vintage, world, champion] |
|AIRLINE BAG VINTAGE JET SET BROWN  |[airline, bag, vintage, jet, set, brown] |
+-----------------------------------+-----------------------------------------+
only showing top 20 rows
```

In [39]:
```
# COMMAND ----------

from pyspark.ml.feature import RegexTokenizer
rt = RegexTokenizer()\
  .setInputCol("Description")\
  .setOutputCol("DescOut")\
  .setPattern(" ")\
  .setGaps(False)\
  .setToLowercase(True)
rt.transform(sales.select("Description")).show(20, False)
```

```
+-----------------------------------+------------------+
|Description                        |DescOut           |
+-----------------------------------+------------------+
|RABBIT NIGHT LIGHT                 |[ ,  ]            |
|DOUGHNUT LIP GLOSS                 |[ ,  ,  ]         |
|12 MESSAGE CARDS WITH ENVELOPES    |[ ,  ,  ,  ]      |
|BLUE HARMONICA IN BOX              |[ ,  ,  ,  ]      |
|GUMBALL COAT RACK                  |[ ,  ]            |
|SKULLS  WATER TRANSFER TATTOOS     |[ ,  ,  ,  ,  ]   |
|FELTCRAFT GIRL AMELIE KIT          |[ ,  ,  ]         |
|CAMOUFLAGE LED TORCH               |[ ,  ]            |
|WHITE SKULL HOT WATER BOTTLE       |[ ,  ,  ,  ,  ]   |
|ENGLISH ROSE HOT WATER BOTTLE      |[ ,  ,  ,  ]      |
|HOT WATER BOTTLE KEEP CALM         |[ ,  ,  ,  ]      |
|SCOTTIE DOG HOT WATER BOTTLE       |[ ,  ,  ,  ]      |
|ROSE CARAVAN DOORSTOP              |[ ,  ]            |
|GINGHAM HEART  DOORSTOP RED        |[ ,  ,  ,  ]      |
|STORAGE TIN VINTAGE LEAF           |[ ,  ,  ]         |
|SET OF 4 KNICK KNACK TINS POPPIES  |[ ,  ,  ,  ,  ,  ]|
|POPCORN HOLDER                     |[ ]               |
|GROW A FLYTRAP OR SUNFLOWER IN TIN |[ ,  ,  ,  ,  ,  ]|
|AIRLINE BAG VINTAGE WORLD CHAMPION |[ ,  ,  ,  ,  ]   |
|AIRLINE BAG VINTAGE JET SET BROWN  |[ ,  ,  ,  ,  ]   |
+-----------------------------------+------------------+
only showing top 20 rows
```

In [40]:
```
# COMMAND ----------

from pyspark.ml.feature import StopWordsRemover
englishStopWords = StopWordsRemover.loadDefaultStopWords("english")
stops = StopWordsRemover()\
  .setStopWords(englishStopWords)\
  .setInputCol("DescOut")
stops.transform(tokenized).show(5, False)
```

```
+------------------------------+--------------------------------------+------------------------------
---------+
|Description                   |DescOut                               |StopWordsRemover_017bc435315
e__output|
+------------------------------+--------------------------------------+------------------------------
---------+
|RABBIT NIGHT LIGHT            |[rabbit, night, light]                |[rabbit, night, light]
|
|DOUGHNUT LIP GLOSS            |[doughnut, lip, gloss]                |[doughnut, lip, gloss]
|
|12 MESSAGE CARDS WITH ENVELOPES|[12, message, cards, with, envelopes]|[12, message, cards, envelop
es]      |
|BLUE HARMONICA IN BOX         |[blue, harmonica, in, box]            |[blue, harmonica, box]
|
|GUMBALL COAT RACK             |[gumball, coat, rack]                 |[gumball, coat, rack]
|
+------------------------------+--------------------------------------+------------------------------
---------+
only showing top 5 rows
```

In [41]:
```
# COMMAND ----------

from pyspark.ml.feature import NGram
unigram = NGram().setInputCol("DescOut").setN(1)
bigram = NGram().setInputCol("DescOut").setN(2)
unigram.transform(tokenized.select("DescOut")).show(5, False)
bigram.transform(tokenized.select("DescOut")).show(5, False)
```

```
+-----------------------------------+-----------------------------------+
|DescOut                            |NGram_c2f2c82ae6e6__output         |
+-----------------------------------+-----------------------------------+
|[rabbit, night, light]             |[rabbit, night, light]             |
|[doughnut, lip, gloss]             |[doughnut, lip, gloss]             |
|[12, message, cards, with, envelopes]|[12, message, cards, with, envelopes]|
|[blue, harmonica, in, box]         |[blue, harmonica, in, box]         |
|[gumball, coat, rack]              |[gumball, coat, rack]              |
+-----------------------------------+-----------------------------------+
only showing top 5 rows

+-----------------------------------+-------------------------------------------------------------+
|DescOut                            |NGram_105dd1df449c__output                                   |
+-----------------------------------+-------------------------------------------------------------+
|[rabbit, night, light]             |[rabbit night, night light]                                  |
|[doughnut, lip, gloss]             |[doughnut lip, lip gloss]                                    |
|[12, message, cards, with, envelopes]|[12 message, message cards, cards with, with envelopes]|
|[blue, harmonica, in, box]         |[blue harmonica, harmonica in, in box]                       |
|[gumball, coat, rack]              |[gumball coat, coat rack]                                    |
+-----------------------------------+-------------------------------------------------------------+
only showing top 5 rows
```

In [42]:
```
# COMMAND ----------

from pyspark.ml.feature import CountVectorizer
cv = CountVectorizer()\
  .setInputCol("DescOut")\
  .setOutputCol("countVec")\
  .setVocabSize(500)\
  .setMinTF(1)\
  .setMinDF(2)
fittedCV = cv.fit(tokenized)
fittedCV.transform(tokenized).show(5, False)
```

```
+-------------------------------+-----------------------------------+-----------------------------
----------+
|Description                    |DescOut                            |countVec
|
+-------------------------------+-----------------------------------+-----------------------------
----------+
|RABBIT NIGHT LIGHT             |[rabbit, night, light]             |(500,[150,185,212],[1.0,1.0,
1.0])     |
|DOUGHNUT LIP GLOSS             |[doughnut, lip, gloss]             |(500,[462,463,492],[1.0,1.0,
1.0])     |
|12 MESSAGE CARDS WITH ENVELOPES|[12, message, cards, with, envelopes]|(500,[35,41,166],[1.0,1.0,1.
0])       |
|BLUE HARMONICA IN BOX          |[blue, harmonica, in, box]         |(500,[10,16,36,352],[1.0,1.
0,1.0,1.0])|
|GUMBALL COAT RACK              |[gumball, coat, rack]              |(500,[228,280,407],[1.0,1.0,
1.0])     |
+-------------------------------+-----------------------------------+-----------------------------
----------+
only showing top 5 rows
```

In [43]:
```
# COMMAND ----------

tfIdfIn = tokenized\
  .where("array_contains(DescOut, 'red')")\
  .select("DescOut")\
  .limit(10)
tfIdfIn.show(10, False)
```

```
+---------------------------------------+
|DescOut                                |
+---------------------------------------+
|[gingham, heart, , doorstop, red]      |
|[red, floral, feltcraft, shoulder, bag]|
|[alarm, clock, bakelike, red]          |
|[pin, cushion, babushka, red]          |
|[red, retrospot, mini, cases]          |
|[red, kitchen, scales]                 |
|[gingham, heart, , doorstop, red]      |
|[large, red, babushka, notebook]       |
|[red, retrospot, oven, glove]          |
|[red, retrospot, plate]                |
+---------------------------------------+
```

In [44]:
```
# COMMAND ----------

from pyspark.ml.feature import HashingTF, IDF
tf = HashingTF()\
  .setInputCol("DescOut")\
  .setOutputCol("TFOut")\
  .setNumFeatures(10000)
idf = IDF()\
  .setInputCol("TFOut")\
  .setOutputCol("IDFOut")\
  .setMinDocFreq(2)
```

In [45]:
```
# COMMAND ----------

idf.fit(tf.transform(tfIdfIn)).transform(tf.transform(tfIdfIn)).show(10, False)
```

```
+-------------------------------------+-------------------------------------------------------------+-
-------------------------------------------------------------------------------------------------
----------------+
|DescOut                              |TFOut                                                        |I
DFOut
|
+-------------------------------------+-------------------------------------------------------------+-
-------------------------------------------------------------------------------------------------
----------------+
|[gingham, heart, , doorstop, red]    |(10000,[3372,4291,4370,6594,9160],[1.0,1.0,1.0,1.0,1.0])|
(10000,[3372,4291,4370,6594,9160],[1.2992829841302609,0.0,1.2992829841302609,1.2992829841302609,1.2
992829841302609])|
|[red, floral, feltcraft, shoulder, bag]|(10000,[155,1152,4291,5981,6756],[1.0,1.0,1.0,1.0,1.0]) |
(10000,[155,1152,4291,5981,6756],[0.0,0.0,0.0,0.0,0.0])
|
|[alarm, clock, bakelike, red]        |(10000,[4291,4852,4995,9668],[1.0,1.0,1.0,1.0])         |
(10000,[4291,4852,4995,9668],[0.0,0.0,0.0,0.0])
|
|[pin, cushion, babushka, red]        |(10000,[4291,5111,5673,7153],[1.0,1.0,1.0,1.0])         |
(10000,[4291,5111,5673,7153],[0.0,0.0,0.0,1.2992829841302609])
|
|[red, retrospot, mini, cases]        |(10000,[547,1576,2591,4291],[1.0,1.0,1.0,1.0])          |
(10000,[547,1576,2591,4291],[0.0,0.0,1.0116009116784799,0.0])
|
|[red, kitchen, scales]               |(10000,[3461,4291,6214],[1.0,1.0,1.0])                  |
(10000,[3461,4291,6214],[0.0,0.0,0.0])
|
|[gingham, heart, , doorstop, red]    |(10000,[3372,4291,4370,6594,9160],[1.0,1.0,1.0,1.0,1.0])|
(10000,[3372,4291,4370,6594,9160],[1.2992829841302609,0.0,1.2992829841302609,1.2992829841302609,1.2
992829841302609])|
|[large, red, babushka, notebook]     |(10000,[2782,2787,4291,7153],[1.0,1.0,1.0,1.0])         |
(10000,[2782,2787,4291,7153],[0.0,0.0,0.0,1.2992829841302609])
|
|[red, retrospot, oven, glove]        |(10000,[302,2591,4291,8242],[1.0,1.0,1.0,1.0])          |
(10000,[302,2591,4291,8242],[0.0,1.0116009116784799,0.0,0.0])
|
|[red, retrospot, plate]              |(10000,[2591,4291,4456],[1.0,1.0,1.0])                  |
(10000,[2591,4291,4456],[1.0116009116784799,0.0,0.0])
|
```

```
+-----------------------------------+--------------------------------------------------------------+-
------------------------------------------------------------------------------------------------
----------------+
```

In [46]:
```
# COMMAND ----------

from pyspark.ml.feature import Word2Vec
# Input data: Each row is a bag of words from a sentence or document.
documentDF = spark.createDataFrame([
    ("Hi I heard about Spark".split(" "), ),
    ("I wish Java could use case classes".split(" "), ),
    ("Logistic regression models are neat".split(" "), )
], ["text"])
# Learn a mapping from words to Vectors.
word2Vec = Word2Vec(vectorSize=3, minCount=0, inputCol="text",
  outputCol="result")
model = word2Vec.fit(documentDF)
result = model.transform(documentDF)
for row in result.collect():
    text, vector = row
    print("Text: [%s] => \nVector: %s\n" % (", ".join(text), str(vector)))
```

```
Text: [Hi, I, heard, about, Spark] =>
Vector: [0.012779767811298371,-0.09340975657105446,-0.10830843970179559]

Text: [I, wish, Java, could, use, case, classes] =>
Vector: [0.07612769335641392,0.03451743721961975,-0.04290600613291774]

Text: [Logistic, regression, models, are, neat] =>
Vector: [-0.06759414225816728,0.045298346877098085,0.05302179120481015]
```

In [47]:
```
# COMMAND ----------

from pyspark.ml.feature import PCA
pca = PCA().setInputCol("features").setK(2)
pca.fit(scaleDF).transform(scaleDF).show(20, False)
```

```
+---+--------------+-----------------------------------------+
|id |features      |PCA_aac2b4c991f1__output                 |
+---+--------------+-----------------------------------------+
|0  |[1.0,0.1,-1.0]|[0.07137194992484153,-0.4526654881478463]|
|1  |[2.0,1.1,1.0] |[-1.6804946984073725,1.2593401322219144] |
|0  |[1.0,0.1,-1.0]|[0.07137194992484153,-0.4526654881478463]|
|1  |[2.0,1.1,1.0] |[-1.6804946984073725,1.2593401322219144] |
|1  |[3.0,10.1,3.0]|[-10.872398139848944,0.030962697060149758]|
+---+--------------+-----------------------------------------+
```

In [48]:
```
# COMMAND ----------

from pyspark.ml.feature import PolynomialExpansion
pe = PolynomialExpansion().setInputCol("features").setDegree(2)
pe.transform(scaleDF).show(5, False)
```

```
+---+--------------+-----------------------------------------------------------------------------
----+
|id |features      |PolynomialExpansion_58cb74d31024__output
|
+---+--------------+-----------------------------------------------------------------------------
----+
|0  |[1.0,0.1,-1.0]|[1.0,1.0,0.1,0.1,0.010000000000000002,-1.0,-1.0,-0.1,1.0]
|
|1  |[2.0,1.1,1.0] |[2.0,4.0,1.1,2.2,1.2100000000000002,1.0,2.0,1.1,1.0]
|
|0  |[1.0,0.1,-1.0]|[1.0,1.0,0.1,0.1,0.010000000000000002,-1.0,-1.0,-0.1,1.0]
|
|1  |[2.0,1.1,1.0] |[2.0,4.0,1.1,2.2,1.2100000000000002,1.0,2.0,1.1,1.0]
|
|1  |[3.0,10.1,3.0]|[3.0,9.0,10.1,30.299999999999997,102.00999999999999,3.0,9.0,30.299999999999997,
9.0]|
+---+--------------+-----------------------------------------------------------------------------
----+
```

In [ ]:
```
# COMMAND ----------

from pyspark.ml.feature import ChiSqSelector, Tokenizer
tkn = Tokenizer().setInputCol("Description").setOutputCol("DescOut")
tokenized = tkn\
  .transform(sales.select("Description", "CustomerId"))\
  .where("CustomerId IS NOT NULL")
prechi = fittedCV.transform(tokenized)\
  .where("CustomerId IS NOT NULL")
chisq = ChiSqSelector()\
  .setFeaturesCol("countVec")\
  .setLabelCol("CustomerId")\
  .setNumTopFeatures(2)
chisq.fit(prechi).transform(prechi)\
  .drop("customerId", "Description", "DescOut").show(5, False)
```

In [ ]:
```
# COMMAND ----------

fittedPCA = pca.fit(scaleDF)
fittedPCA.write().overwrite().save("/tmp/fittedPCA")
```

In [ ]:
```
# COMMAND ----------

from pyspark.ml.feature import PCAModel
loadedPCA = PCAModel.load("/tmp/fittedPCA")
loadedPCA.transform(scaleDF).show(5, False)


# COMMAND ----------
```

In [ ]:
```

```