In [2]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-06-types")\
    .getOrCreate()

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

In [3]:
```
file_path = SPARK_BOOK_DATA_PATH + "/data/retail-data/by-day/2010-12-01
df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load(file_path)
```

In [4]:
```
df.printSchema()
df.createOrReplaceTempView("dfTable")
```

```
root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)
```

In [5]:
```
# COMMAND ----------

# from pyspark.sql.functions import lit
df.select(F.lit(5), F.lit("five"), F.lit(5.0))
```

Out[5]:
```
DataFrame[5: int, five: string, 5.0: double]
```

In [6]:
```
# COMMAND ----------

# from pyspark.sql.functions import col
df.where(F.col("InvoiceNo") != 536365)\
  .select("InvoiceNo", "Description")\
  .show(5, False)
```

```
+---------+-----------------------------+
|InvoiceNo|Description                  |
+---------+-----------------------------+
|536366   |HAND WARMER UNION JACK       |
|536366   |HAND WARMER RED POLKA DOT    |
|536367   |ASSORTED COLOUR BIRD ORNAMENT|
|536367   |POPPY'S PLAYHOUSE BEDROOM    |
|536367   |POPPY'S PLAYHOUSE KITCHEN    |
+---------+-----------------------------+
only showing top 5 rows
```

In [8]:
```
# COMMAND ----------

# from pyspark.sql.functions import instr
priceFilter = F.col("UnitPrice") > 600
descripFilter = F.instr(df.Description, "POSTAGE") >= 1
df.where(df.StockCode.isin("DOT"))\
    .where(priceFilter | descripFilter)\
    .show(5)
```

```
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+
|InvoiceNo|StockCode|   Description|Quantity|        InvoiceDate|UnitP
rice|CustomerID|       Country|
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+
|   536544|      DOT|DOTCOM POSTAGE|       1|2010-12-01 14:32:00|   56
9.77|      null|United Kingdom|
|   536592|      DOT|DOTCOM POSTAGE|       1|2010-12-01 17:06:00|   60
7.49|      null|United Kingdom|
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+
```

In [9]:
```
# COMMAND ----------

# from pyspark.sql.functions import instr
DOTCodeFilter = F.col("StockCode") == "DOT"
priceFilter = F.col("UnitPrice") > 600
descripFilter = F.instr(F.col("Description"), "POSTAGE") >= 1
df.withColumn("isExpensive", DOTCodeFilter & (priceFilter | descripFilt
    .where("isExpensive")\
    .select("InvoiceNo", "unitPrice", "isExpensive")\
    .show(5)
```

```
+---------+---------+-----------+
|InvoiceNo|unitPrice|isExpensive|
+---------+---------+-----------+
|   536544|   569.77|       true|
|   536592|   607.49|       true|
+---------+---------+-----------+
```

In [10]:
```
# COMMAND ----------

# from pyspark.sql.functions import expr
df.withColumn("isExpensive", F.expr("NOT UnitPrice <= 250"))\
  .where("isExpensive")\
  .select("InvoiceNo", "Description", "UnitPrice", "isExpensive")\
  .show(5)
```

```
+---------+--------------+---------+-----------+
|InvoiceNo|   Description|UnitPrice|isExpensive|
+---------+--------------+---------+-----------+
|   536544|DOTCOM POSTAGE|   569.77|       true|
|   536592|DOTCOM POSTAGE|   607.49|       true|
+---------+--------------+---------+-----------+
```

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

# from pyspark.sql.functions import expr, pow
fabricatedQuantity = F.pow(F.col("Quantity") * F.col("UnitPrice"), 2) +
df.select(F.expr("CustomerId"), fabricatedQuantity.alias("realQuantity"
    .show(2)
```

```
+----------+------------------+
|CustomerId|      realQuantity|
+----------+------------------+
|   17850.0|239.08999999999997|
|   17850.0|          418.7156|
+----------+------------------+
only showing top 2 rows
```

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

df.selectExpr(
  "CustomerId",
  "(POWER((Quantity * UnitPrice), 2.0) + 5) as realQuantity")\
    .show(2)
```

```
+----------+------------------+
|CustomerId|      realQuantity|
+----------+------------------+
|   17850.0|239.08999999999997|
|   17850.0|          418.7156|
+----------+------------------+
only showing top 2 rows
```

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

# from pyspark.sql.functions import lit, round, bround

df.select(F.round(F.lit("2.5")), F.bround(F.lit("2.5")))\
    .show(2)
```

```
+-------------+-------------+
|round(2.5, 0)|bround(2.5, 0)|
+-------------+-------------+
|          3.0|           2.0|
|          3.0|           2.0|
+-------------+-------------+
only showing top 2 rows
```

In [15]:
```
# COMMAND ----------

# from pyspark.sql.functions import corr
df.stat.corr("Quantity", "UnitPrice")
```

Out[15]:
```
-0.04112314436835551
```

In [16]:
```
df.select(F.corr("Quantity", "UnitPrice"))\
    .show()
```

```
+-------------------------+
|corr(Quantity, UnitPrice)|
+-------------------------+
|     -0.04112314436835551|
+-------------------------+
```

In [17]:
```
df.show(5)
```

```
+---------+---------+--------------------+--------+--------------------
+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|         InvoiceDate
|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+--------------------
+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|2010-12-01 08:26:00
|     2.55|   17850.0|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|2010-12-01 08:26:00
|     2.75|   17850.0|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|
+---------+---------+--------------------+--------+--------------------
+---------+----------+--------------+
only showing top 5 rows
```

In [18]:
```
# COMMAND ----------

df.describe().show()
```

```
+-------+-----------------+------------------+------------------+--------------------+---
---------------+------------------+------------------+--------------+
|summary|        InvoiceNo|         StockCode|         Description|
Quantity|         UnitPrice|        CustomerID|       Country|
+-------+-----------------+------------------+------------------+--------------------+---
---------------+------------------+------------------+--------------+
|  count|             3108|              3108|                3098|
3108|              3108|              1968|          3108|
|   mean| 536516.684944841|27834.304044117645|                null| 8.
627413127413128| 4.151946589446603|15661.388719512195|          null|
| stddev|72.89447869788873|17407.897548583845|                null|26.
371821677029203|15.638659854603892|1854.4496996893627|          null|
|    min|           536365|             10002| 4 PURPLE FLOCK D...|
-24|              0.0|           12431.0|     Australia|
|    max|          C536548|              POST|ZINC WILLIE WINKI...|
600|            607.49|           18229.0|United Kingdom|
+-------+-----------------+------------------+------------------+--------------------+---
---------------+------------------+------------------+--------------+
```

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

# from pyspark.sql.functions import count, mean, stddev_pop, min, max


# COMMAND ----------

colName = "UnitPrice"
quantileProbs = [0.5]
relError = 0.05
df.stat.approxQuantile("UnitPrice", quantileProbs, relError) # 2.51
```

Out[19]:
```
[2.51]
```

In [20]:
```
# COMMAND ----------

df.stat.crosstab("StockCode", "Quantity").show()
```

```
+-----------------+---+---+---+---+---+---+---+---+---+---+---+---+---+--
-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--
-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+
|StockCode_Quantity| -1|-10|-12| -2|-24| -3| -4| -5| -6| -7|  1| 10|10
0| 11| 12|120|128| 13| 14|144| 15| 16| 17| 18| 19|192|  2| 20|200| 21|
216| 22| 23| 24| 25|252| 27| 28|288|  3| 30| 32| 33| 34| 36|384|  4| 4
0|432| 47| 48|480|  5| 50| 56|  6| 60|600| 64|  7| 70| 72|  8| 80|  9|
96|
+-----------------+---+---+---+---+---+---+---+---+---+---+---+---+---+--
-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--
-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+
|             22578|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             21327|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  2|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             22064|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             21080|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|
|             22219|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  3|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             21908|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             22818|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|            15056BL|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             72817|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|             22545|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
```

```
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              22988|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|
|              22274|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              20750|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  2|  1|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              82616C|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              21703|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              22899|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  2|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|
|              22379|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  1|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  2|  1|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              22422|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  2|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              22769|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|              22585|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  1|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
+------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+--
-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--
-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
---+
only showing top 20 rows
```

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

df.stat.freqItems(["StockCode", "Quantity"]).show()
```

```
+--------------------+--------------------+
| StockCode_freqItems|  Quantity_freqItems|
+--------------------+--------------------+
|[90214E, 20728, 2...|[200, 128, 23, 32...|
+--------------------+--------------------+
```

In [22]:
```
# COMMAND ----------

# from pyspark.sql.functions import monotonically_increasing_id

df.select(F.monotonically_increasing_id()).show(10)
```

```
+-----------------------------+
|monotonically_increasing_id()|
+-----------------------------+
|                            0|
|                            1|
|                            2|
|                            3|
|                            4|
|                            5|
|                            6|
|                            7|
|                            8|
|                            9|
+-----------------------------+
only showing top 10 rows
```

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

# from pyspark.sql.functions import monotonically_increasing_id

# add Id column
df = df.withColumn("Id", F.monotonically_increasing_id())

df.show(10)
```

```
+---------+---------+--------------------+--------+--------------------
+---------+----------+--------------+---+
|InvoiceNo|StockCode|         Description|Quantity|        InvoiceDate
|UnitPrice|CustomerID|       Country| Id|
+---------+---------+--------------------+--------+--------------------
+---------+----------+--------------+---+
|   536365|   85123A|WHITE HANGING HEA...|       6|2010-12-01 08:26:00
|     2.55|   17850.0|United Kingdom|  0|
|   536365|    71053| WHITE METAL LANTERN|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|  1|
|   536365|   84406B|CREAM CUPID HEART...|       8|2010-12-01 08:26:00
|     2.75|   17850.0|United Kingdom|  2|
|   536365|   84029G|KNITTED UNION FLA...|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|  3|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|  4|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|2010-12-01 08:26:00
|     7.65|   17850.0|United Kingdom|  5|
|   536365|    21730|GLASS STAR FROSTE...|       6|2010-12-01 08:26:00
|     4.25|   17850.0|United Kingdom|  6|
|   536366|    22633|HAND WARMER UNION...|       6|2010-12-01 08:28:00
|     1.85|   17850.0|United Kingdom|  7|
|   536366|    22632|HAND WARMER RED P...|       6|2010-12-01 08:28:00
|     1.85|   17850.0|United Kingdom|  8|
|   536367|    84879|ASSORTED COLOUR B...|      32|2010-12-01 08:34:00
|     1.69|   13047.0|United Kingdom|  9|
+---------+---------+--------------------+--------+--------------------
+---------+----------+--------------+---+
only showing top 10 rows
```

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

# from pyspark.sql.functions import initcap
df.select(F.initcap(F.col("Description"))).show(5, False)  # False - di
```

```
+-----------------------------------+
|initcap(Description)               |
+-----------------------------------+
|White Hanging Heart T-light Holder |
|White Metal Lantern                |
|Cream Cupid Hearts Coat Hanger     |
|Knitted Union Flag Hot Water Bottle|
|Red Woolly Hottie White Heart.     |
+-----------------------------------+
only showing top 5 rows
```

In [32]:
```
# COMMAND ----------

# from pyspark.sql.functions import lower, upper
df.select(F.col("Description"),
    F.lower(F.col("Description")),
    F.upper(F.col("Description")))\
    .show(2, False)
```

```
+-----------------------------------+-----------------------------------
+-----------------------------------+
|Description                        |lower(Description)
|upper(Description)                 |
+-----------------------------------+-----------------------------------
+-----------------------------------+
|WHITE HANGING HEART T-LIGHT HOLDER|white hanging heart t-light holder
|WHITE HANGING HEART T-LIGHT HOLDER|
|WHITE METAL LANTERN                |white metal lantern
|WHITE METAL LANTERN                |
+-----------------------------------+-----------------------------------
+-----------------------------------+
only showing top 2 rows
```

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

# from pyspark.sql.functions import lit, ltrim, rtrim, rpad, lpad, trim
df.select(
    F.ltrim(F.lit("    HELLO    ")).alias("ltrim"),
    F.rtrim(F.lit("    HELLO    ")).alias("rtrim"),
    F.trim(F.lit("    HELLO    ")).alias("trim"),
    F.lpad(F.lit("HELLO"), 3, " ").alias("lp"),
    F.rpad(F.lit("HELLO"), 10, " ").alias("rp"))\
    .show(2)
```

```
+---------+---------+-----+---+----------+
|    ltrim|    rtrim| trim| lp|        rp|
+---------+---------+-----+---+----------+
|HELLO    |    HELLO|HELLO|HEL|HELLO     |
|HELLO    |    HELLO|HELLO|HEL|HELLO     |
+---------+---------+-----+---+----------+
only showing top 2 rows
```

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

# from pyspark.sql.functions import translate
df.select(F.translate(F.col("Description"), "LEET", "1337"), F.col("Des
  .show(2,False)
```

```
+---------------------------------+---------------------------------
+
|translate(Description, LEET, 1337)|Description
|
+---------------------------------+---------------------------------
+
|WHI73 HANGING H3AR7 7-1IGH7 HO1D3R|WHITE HANGING HEART T-LIGHT HOLDER
|
|WHI73 M37A1 1AN73RN               |WHITE METAL LANTERN
|
+---------------------------------+---------------------------------
+
only showing top 2 rows
```

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

# from pyspark.sql.functions import regexp_replace

regex_string = "BLACK|WHITE|RED|GREEN|BLUE"
df.select(
    F.regexp_replace(F.col("Description"), regex_string, "COLOR").alias
    F.col("Description"))\
    .show(2, False)
```

```
+----------------------------------+----------------------------------
+
|color_clean                       |Description
|
+----------------------------------+----------------------------------
+
|COLOR HANGING HEART T-LIGHT HOLDER|WHITE HANGING HEART T-LIGHT HOLDER
|
|COLOR METAL LANTERN               |WHITE METAL LANTERN
|
+----------------------------------+----------------------------------
+
only showing top 2 rows
```

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

# from pyspark.sql.functions import regexp_extract
extract_str = "(BLACK|WHITE|RED|GREEN|BLUE)"
df.select(
     F.regexp_extract(F.col("Description"), extract_str, 1).alias("colo
     F.col("Description"))\
    .show(2,False)
```

```
+-----------+---------------------------------+
|color_clean|Description                      |
+-----------+---------------------------------+
|WHITE      |WHITE HANGING HEART T-LIGHT HOLDER|
|WHITE      |WHITE METAL LANTERN              |
+-----------+---------------------------------+
only showing top 2 rows
```

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

# from pyspark.sql.functions import instr
containsBlack = F.instr(F.col("Description"), "BLACK") >= 1
containsWhite = F.instr(F.col("Description"), "WHITE") >= 1
df.withColumn("hasSimpleColor", containsBlack | containsWhite)\
  .where("hasSimpleColor")\
  .select("Description")\
    .show(3, False)
```

```
+---------------------------------+
|Description                      |
+---------------------------------+
|WHITE HANGING HEART T-LIGHT HOLDER|
|WHITE METAL LANTERN              |
|RED WOOLLY HOTTIE WHITE HEART.   |
+---------------------------------+
only showing top 3 rows
```

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

# from pyspark.sql.functions import expr, locate
simpleColors = ["black", "white", "red", "green", "blue"]
def color_locator(column, color_string):
  return locate(color_string.upper(), column)\
          .cast("boolean")\
          .alias("is_" + color_string)
selectedColumns = [color_locator(df.Description, c) for c in simpleColo
selectedColumns.append(expr("*")) # has to a be Column type

df.select(*selectedColumns).where(expr("is_white OR is_red"))\
  .select("Description")\
    .show(3, False)
```

```
+---------------------------------+
|Description                      |
+---------------------------------+
|WHITE HANGING HEART T-LIGHT HOLDER|
|WHITE METAL LANTERN              |
|RED WOOLLY HOTTIE WHITE HEART.   |
+---------------------------------+
only showing top 3 rows
```

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

# from pyspark.sql.functions import current_date, current_timestamp

dateDF = spark.range(10)\
  .withColumn("today", F.current_date())\
  .withColumn("now", F.current_timestamp())

dateDF.createOrReplaceTempView("dateTable")

dateDF.show(4, False)
```

```
+---+----------+----------------------+
|id |today     |now                   |
+---+----------+----------------------+
|0  |2019-09-28|2019-09-28 16:48:35.318|
|1  |2019-09-28|2019-09-28 16:48:35.318|
|2  |2019-09-28|2019-09-28 16:48:35.318|
|3  |2019-09-28|2019-09-28 16:48:35.318|
+---+----------+----------------------+
only showing top 4 rows
```

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

# from pyspark.sql.functions import date_add, date_sub

dateDF.select(F.date_sub(F.col("today"), 5), F.date_add(F.col("today"),
    .show(1)
```

```
+------------------+------------------+
|date_sub(today, 5)|date_add(today, 5)|
+------------------+------------------+
|        2019-09-23|        2019-10-03|
+------------------+------------------+
only showing top 1 row
```

In [51]:
```
# COMMAND ----------

# from pyspark.sql.functions import datediff, months_between, to_date

dateDF.withColumn("week_ago", F.date_sub(F.col("today"), 7))\
    .select(F.datediff(F.col("week_ago"), F.col("today")))\
    .show(1)
```

```
+-------------------------+
|datediff(week_ago, today)|
+-------------------------+
|                       -7|
+-------------------------+
only showing top 1 row
```

In [53]:
```
dateDF.select(
    F.to_date(F.lit("2016-01-01")).alias("start"),
    F.to_date(F.lit("2017-05-22")).alias("end"))\
    .select(F.months_between(F.col("start"), F.col("end")))\
    .show(1)
```

```
+--------------------------------+
|months_between(start, end, true)|
+--------------------------------+
|                     -16.67741935|
+--------------------------------+
only showing top 1 row
```

In [54]:
```
# COMMAND ----------

# from pyspark.sql.functions import to_date, lit
spark.range(5)\
    .withColumn("date", F.lit("2017-01-01"))\
    .select(F.to_date(F.col("date")))\
    .show(1)
```

```
+---------------+
|to_date(`date`)|
+---------------+
|     2017-01-01|
+---------------+
only showing top 1 row
```

In [56]:
```
# COMMAND ----------

# from pyspark.sql.functions import to_date
dateFormat = "yyyy-dd-MM"
cleanDateDF = spark.range(1).select(
    F.to_date(F.lit("2017-12-11"), dateFormat).alias("date"),
    F.to_date(F.lit("2017-20-12"), dateFormat).alias("date2"))
cleanDateDF.createOrReplaceTempView("dateTable2")
cleanDateDF.show(5)
```

```
+----------+----------+
|      date|     date2|
+----------+----------+
|2017-11-12|2017-12-20|
+----------+----------+
```

In [57]:
```
# COMMAND ----------

# from pyspark.sql.functions import to_timestamp
cleanDateDF.select(F.to_timestamp(F.col("date"), dateFormat))\
    .show()
```

```
+----------------------------------+
|to_timestamp(`date`, 'yyyy-dd-MM')|
+----------------------------------+
|               2017-11-12 00:00:00|
+----------------------------------+
```

In [63]:
```
# COMMAND ----------

# from pyspark.sql.functions import coalesce


df.select(F.col("Description"), F.col("CustomerId"))\
    .show(10, False)
```

```
+-----------------------------------+----------+
|Description                        |CustomerId|
+-----------------------------------+----------+
|WHITE HANGING HEART T-LIGHT HOLDER |17850.0   |
|WHITE METAL LANTERN                |17850.0   |
|CREAM CUPID HEARTS COAT HANGER     |17850.0   |
|KNITTED UNION FLAG HOT WATER BOTTLE|17850.0   |
|RED WOOLLY HOTTIE WHITE HEART.     |17850.0   |
|SET 7 BABUSHKA NESTING BOXES       |17850.0   |
|GLASS STAR FROSTED T-LIGHT HOLDER  |17850.0   |
|HAND WARMER UNION JACK             |17850.0   |
|HAND WARMER RED POLKA DOT          |17850.0   |
|ASSORTED COLOUR BIRD ORNAMENT      |13047.0   |
+-----------------------------------+----------+
only showing top 10 rows
```

In [64]:
```
# COMMAND ----------

df.na.drop("all", subset=["StockCode", "InvoiceNo"])
```

Out[64]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: timestamp, UnitPrice: double, CustomerID: d
ouble, Country: string, Id: bigint]
```

In [65]:
```
# COMMAND ----------

df.na.fill("all", subset=["StockCode", "InvoiceNo"])
```

Out[65]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: timestamp, UnitPrice: double, CustomerID: d
ouble, Country: string, Id: bigint]
```

In [66]:
```
# COMMAND ----------

fill_cols_vals = {"StockCode": 5, "Description" : "No Value"}
df.na.fill(fill_cols_vals)
```

Out[66]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: timestamp, UnitPrice: double, CustomerID: d
ouble, Country: string, Id: bigint]
```

In [67]:
```
# COMMAND ----------

df.na.replace([""], ["UNKNOWN"], "Description")
```

Out[67]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: timestamp, UnitPrice: double, CustomerID: d
ouble, Country: string, Id: bigint]
```

In [68]:
```
# COMMAND ----------

# from pyspark.sql.functions import struct
complexDF = df.select(F.struct("Description", "InvoiceNo").alias("compl

complexDF.createOrReplaceTempView("complexDF")
```

In [72]:
```
# COMMAND ----------

# from pyspark.sql.functions import split
df.select(F.split(F.col("Description"), " ")).show(2, False)
```

```
+-----------------------------------------+
|split(Description,  )                    |
+-----------------------------------------+
|[WHITE, HANGING, HEART, T-LIGHT, HOLDER]|
|[WHITE, METAL, LANTERN]                  |
+-----------------------------------------+
only showing top 2 rows
```

In [73]:
```
# COMMAND ----------

df.select(F.split(F.col("Description"), " ").alias("array_col"))\
    .selectExpr("array_col[0]")\
    .show(2)
```

```
+------------+
|array_col[0]|
+------------+
|       WHITE|
|       WHITE|
+------------+
only showing top 2 rows
```

In [77]:
```
# COMMAND ----------

# from pyspark.sql.functions import size
df.select("Description",
    F.size(F.split(F.col("Description"), " ")))\
    .show(2, False) # shows 5 and 3
```

```
+---------------------------------+--------------------------+
|Description                      |size(split(Description,  ))|
+---------------------------------+--------------------------+
|WHITE HANGING HEART T-LIGHT HOLDER|5                         |
|WHITE METAL LANTERN              |3                         |
+---------------------------------+--------------------------+
only showing top 2 rows
```

In [78]:
```
# COMMAND ----------

# from pyspark.sql.functions import array_contains
df.select(F.array_contains(F.split(F.col("Description"), " "), "WHITE")
```

```
+-------------------------------------------+
|array_contains(split(Description,  ), WHITE)|
+-------------------------------------------+
|                                       true|
|                                       true|
+-------------------------------------------+
only showing top 2 rows
```

In [81]:
```
# COMMAND ----------

# from pyspark.sql.functions import split, explode

df.withColumn("splitted", F.split(F.col("Description"), " "))\
  .withColumn("exploded", F.explode(F.col("splitted")))\
  .select("Description", "InvoiceNo", "exploded")\
  .show(2, False)
```

```
+---------------------------------+---------+--------+
|Description                      |InvoiceNo|exploded|
+---------------------------------+---------+--------+
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |WHITE   |
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |HANGING |
+---------------------------------+---------+--------+
only showing top 2 rows
```

In [83]:
```
# COMMAND ----------

# from pyspark.sql.functions import create_map
df.select(F.create_map(F.col("Description"), F.col("InvoiceNo")).alias(
  .show(2, False)
```

```
+--------------------------------------------+
|complex_map                                 |
+--------------------------------------------+
|[WHITE HANGING HEART T-LIGHT HOLDER -> 536365]|
|[WHITE METAL LANTERN -> 536365]             |
+--------------------------------------------+
only showing top 2 rows
```

In [87]:
```
# COMMAND ----------

df.select(F.create_map(F.col("Description"), F.col("InvoiceNo")).alias(
  .selectExpr("complex_map['WHITE METAL LANTERN']")\
    .show(2)
```

```
+--------------------------------+
|complex_map[WHITE METAL LANTERN]|
+--------------------------------+
|                            null|
|                          536365|
+--------------------------------+
only showing top 2 rows
```

In [93]:
```
df.show(2)
```

```
+---------+---------+--------------------+--------+-------------------
+---------+----------+--------------+---+
|InvoiceNo|StockCode|         Description|Quantity|        InvoiceDate
|UnitPrice|CustomerID|       Country| Id|
+---------+---------+--------------------+--------+-------------------
+---------+----------+--------------+---+
|   536365|   85123A|WHITE HANGING HEA...|       6|2010-12-01 08:26:00
|     2.55|   17850.0|United Kingdom|  0|
|   536365|    71053| WHITE METAL LANTERN|       6|2010-12-01 08:26:00
|     3.39|   17850.0|United Kingdom|  1|
+---------+---------+--------------------+--------+-------------------
+---------+----------+--------------+---+
only showing top 2 rows
```

In [94]:
```
# COMMAND ----------

df.select(
    F.create_map(F.col("Description"), F.col("InvoiceNo")).alias("compl
    .selectExpr("explode(complex_map)")\
    .show(2, False)
```

```
+---------------------------------+------+
|key                              |value |
+---------------------------------+------+
|WHITE HANGING HEART T-LIGHT HOLDER|536365|
|WHITE METAL LANTERN              |536365|
+---------------------------------+------+
only showing top 2 rows
```

In [95]:
```
# COMMAND ----------

jsonDF = spark.range(1).selectExpr("""
  '{"myJSONKey" : {"myJSONValue" : [1, 2, 3]}}' as jsonString""")


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

In [99]:
```
jsonDF.show(2, False)
```

```
+-----------------------------------------+
|jsonString                               |
+-----------------------------------------+
|{"myJSONKey" : {"myJSONValue" : [1, 2, 3]}}|
+-----------------------------------------+
```

In [101]:
```
# from pyspark.sql.functions import get_json_object, json_tuple

jsonDF.select(
    F.get_json_object(F.col("jsonString"), "$.myJSONKey.myJSONValue[1]"
    F.json_tuple(F.col("jsonString"), "myJSONKey"))\
    .show(2, False)
```

```
+------+----------------------+
|column|c0                    |
+------+----------------------+
|2     |{"myJSONValue":[1,2,3]}|
+------+----------------------+
```

In [103]:
```
# COMMAND ----------

# from pyspark.sql.functions import to_json
df.selectExpr("(InvoiceNo, Description) as myStruct")\
  .select(F.to_json(F.col("myStruct")))\
    .show(3, False)
```

```
+-----------------------------------------------------------------
----+
|structstojson(myStruct)
|
+-----------------------------------------------------------------
----+
|{"InvoiceNo":"536365","Description":"WHITE HANGING HEART T-LIGHT HOLD
ER"}|
|{"InvoiceNo":"536365","Description":"WHITE METAL LANTERN"}
|
|{"InvoiceNo":"536365","Description":"CREAM CUPID HEARTS COAT HANGER"}
|
+-----------------------------------------------------------------
----+
only showing top 3 rows
```

In [105]:
```
# COMMAND ----------

# from pyspark.sql.functions import from_json
# from pyspark.sql.types import *

parseSchema = StructType((
  StructField("InvoiceNo",StringType(),True),
  StructField("Description",StringType(),True)))

df.selectExpr("(InvoiceNo, Description) as myStruct")\
  .select(F.to_json(F.col("myStruct")).alias("newJSON"))\
  .select(F.from_json(F.col("newJSON"), parseSchema), F.col("newJSON"))\
    .show(2, False)
```

```
+-----------------------------------------+----------------------
-----------------------------------------------+
|jsontostructs(newJSON)                   |newJSON
|
+-----------------------------------------+----------------------
-----------------------------------------------+
|[536365, WHITE HANGING HEART T-LIGHT HOLDER]|{"InvoiceNo":"536365","D
escription":"WHITE HANGING HEART T-LIGHT HOLDER"}|
|[536365, WHITE METAL LANTERN]            |{"InvoiceNo":"536365","D
escription":"WHITE METAL LANTERN"}              |
+-----------------------------------------+----------------------
-----------------------------------------------+
only showing top 2 rows
```

In [107]:
```
# COMMAND ----------

udfExampleDF = spark.range(5).toDF("num")
```

In [115]:
```
def power3(double_value):
  return float(double_value ** 3)
power3(2.0)
```

Out[115]:
```
8.0
```

In [116]:
```
# COMMAND ----------

# from pyspark.sql.functions import udf
power3udf = F.udf(power3)
```

In [117]:
```
# COMMAND ----------

# from pyspark.sql.functions import col
udfExampleDF.select("num", power3udf(F.col("num")).alias("num_cubed"))\
    .show(6)
```

```
+---+---------+
|num|num_cubed|
+---+---------+
|  0|      0.0|
|  1|      1.0|
|  2|      8.0|
|  3|     27.0|
|  4|     64.0|
+---+---------+
```

In [118]:
```
# COMMAND ----------

# from pyspark.sql.types import IntegerType, DoubleType
spark.udf.register("power3py", power3, DoubleType())
```

Out[118]:
```
<function __main__.power3(double_value)>
```

In [120]:
```
# COMMAND ----------

udfExampleDF.selectExpr("power3py(num)").show(5)
# registered via Python
```

```
+-------------+
|power3py(num)|
+-------------+
|          0.0|
|          1.0|
|          8.0|
|         27.0|
|         64.0|
+-------------+
```

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

In [111]:
```
spark.sql("show user functions like 'power*'").show()
```

```
+--------+
|function|
+--------+
+--------+
```