- pyspark.sql.types (https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#module-pyspark.sql.types)
- pyspark.sql.functions (https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#module-pyspark.sql.functions)

Other examples

- Spark Data Operations (https://github.com/PacktPublishing/Mastering-Big-Data-Analytics-with-PySpark/blob/master/Section%202%20-%20Working%20with%20PySpark/2.5/2.5%20-%20Spark%20Data%20Operations.ipynb)

In [1]:
```
from IPython.display import display, clear_output
```

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
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: string (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)
```

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

```
+---------+---------+-----------------------------------+--------+----
--------------+---------+----------+--------------+
|InvoiceNo|StockCode|Description                        |Quantity|Invo
iceDate       |UnitPrice|CustomerID|Country       |
+---------+---------+-----------------------------------+--------+----
--------------+---------+----------+--------------+
|536365   |85123A   |WHITE HANGING HEART T-LIGHT HOLDER |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 HEARTS COAT HANGER     |8       |2010
-12-01 08:26:00|2.75     |17850.0   |United Kingdom|
|536365   |84029G   |KNITTED UNION FLAG HOT WATER BOTTLE|6       |2010
-12-01 08:26:00|3.39     |17850.0   |United Kingdom|
|536365   |84029E   |RED WOOLLY HOTTIE WHITE HEART.     |6       |2010
-12-01 08:26:00|3.39     |17850.0   |United Kingdom|
|536365   |22752    |SET 7 BABUSHKA NESTING BOXES       |2       |2010
-12-01 08:26:00|7.65     |17850.0   |United Kingdom|
|536365   |21730    |GLASS STAR FROSTED T-LIGHT HOLDER  |6       |2010
-12-01 08:26:00|4.25     |17850.0   |United Kingdom|
|536366   |22633    |HAND WARMER UNION JACK             |6       |2010
-12-01 08:28:00|1.85     |17850.0   |United Kingdom|
|536366   |22632    |HAND WARMER RED POLKA DOT          |6       |2010
-12-01 08:28:00|1.85     |17850.0   |United Kingdom|
|536367   |84879    |ASSORTED COLOUR BIRD ORNAMENT      |32      |2010
-12-01 08:34:00|1.69     |13047.0   |United Kingdom|
+---------+---------+-----------------------------------+--------+----
--------------+---------+----------+--------------+
only showing top 10 rows
```

In [6]:
```python
df.count()
```

Out[6]:
```
3108
```

In [7]:
```python
spark.sql("select count(*) from dfTable").show()
```

```
+--------+
|count(1)|
+--------+
|    3108|
+--------+
```

lit()

In [8]:
```
df.select(F.lit(5), F.lit("five"), F.lit(5.0)).show(5)
```

```
+---+----+---+
|  5|five|5.0|
+---+----+---+
|  5|five|5.0|
|  5|five|5.0|
|  5|five|5.0|
|  5|five|5.0|
|  5|five|5.0|
+---+----+---+
only showing top 5 rows
```

In [9]:
```
df.select(F.round(F.lit("2.515"), 2), F.bround(F.lit("2.5"))).show(2
```

```
+--------------+-------------+
|round(2.515, 2)|bround(2.5, 0)|
+--------------+-------------+
|          2.52|          2.0|
|          2.52|          2.0|
+--------------+-------------+
only showing top 2 rows
```

In [10]:
```
df.where(F.col("InvoiceNo") != 536365).select("InvoiceNo", "Descript
```

```
+---------+-----------------------------+
|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 [11]:
```
# complex filter
priceFilter = F.col("UnitPrice") > 600
descripFilter = F.instr(df.Description, "POSTAGE") > 0
df.where(df.StockCode.isin("DOT") & (priceFilter | descripFilter)).s
```

```
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+
|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 [12]:
```
DOTCodeFilter = F.col("StockCode") == "DOT"
priceFilter = F.col("UnitPrice") > 600
descripFilter = F.instr(F.col("Description"), "POSTAGE") >= 1
df2 = (df.withColumn("isExpensive", DOTCodeFilter & (priceFilter | d
    .where("isExpensive")
    .select("*")
      )
df2.show(5)
```

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

In [15]:
```
df3 = df2.withColumn("below600", F.expr("UnitPrice < 600")).select('

df3.show(5)
```

```
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+-----------+--------+
|InvoiceNo|StockCode|   Description|Quantity|        InvoiceDate|UnitP
rice|CustomerID|       Country|isExpensive|below600|
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+-----------+--------+
|   536544|      DOT|DOTCOM POSTAGE|       1|2010-12-01 14:32:00|   56
9.77|      null|United Kingdom|       true|    true|
|   536592|      DOT|DOTCOM POSTAGE|       1|2010-12-01 17:06:00|   60
7.49|      null|United Kingdom|       true|   false|
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+-----------+--------+
```

In [16]:
```
df3.where(F.col("isExpensive") & F.col("below600")).select("*").show
```

```
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+-----------+--------+
|InvoiceNo|StockCode|   Description|Quantity|        InvoiceDate|UnitP
rice|CustomerID|       Country|isExpensive|below600|
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+-----------+--------+
|   536544|      DOT|DOTCOM POSTAGE|       1|2010-12-01 14:32:00|   56
9.77|      null|United Kingdom|       true|    true|
+---------+---------+--------------+--------+-------------------+-----
----+----------+--------------+-----------+--------+
```

In [17]:
```
fabricatedQuantity = F.pow(F.col("Quantity") * F.col("UnitPrice"), 2
(df.select("CustomerId", "Quantity", "UnitPrice",
           fabricatedQuantity.alias("fakeQuantity"))
    .show(2))
```

```
+----------+--------+---------+------------------+
|CustomerId|Quantity|UnitPrice|      fakeQuantity|
+----------+--------+---------+------------------+
|   17850.0|       6|     2.55|239.08999999999997|
|   17850.0|       6|     3.39|          418.7156|
+----------+--------+---------+------------------+
only showing top 2 rows
```

In [18]:
```
df.selectExpr(
    "CustomerId",
    "Quantity",
    "UnitPrice",
    "(POWER((Quantity * UnitPrice), 2.0) + 5) as fakeQuantity"
).show(2)
```

```
+----------+--------+---------+-----------------+
|CustomerId|Quantity|UnitPrice|     fakeQuantity|
+----------+--------+---------+-----------------+
|   17850.0|       6|     2.55|239.08999999999997|
|   17850.0|       6|     3.39|         418.7156|
+----------+--------+---------+-----------------+
only showing top 2 rows
```

In [19]:
```
sql_stmt = """
select
    CustomerId,
    Quantity,
    UnitPrice,
    (POWER((Quantity * UnitPrice), 2.0) + 5) as fakeQuantity
from
    dfTable
"""
spark.sql(sql_stmt).show(2)
```

```
+----------+--------+---------+-----------------+
|CustomerId|Quantity|UnitPrice|     fakeQuantity|
+----------+--------+---------+-----------------+
|   17850.0|       6|     2.55|239.08999999999997|
|   17850.0|       6|     3.39|         418.7156|
+----------+--------+---------+-----------------+
only showing top 2 rows
```

describe()

In [20]:
```
display(df.describe().toPandas())
```

	summary	InvoiceNo	StockCode	Description	Quantity	InvoiceD
0	count	3108	3108	3098	3108	31
1	mean	536516.684944841	27834.304044117645	None	8.627413127413128	No
2	stddev	72.89447869788873	17407.897548583845	None	26.371821677029203	No
3	min	536365	10002	4 PURPLE FLOCK DINNER CANDLES	-24	2010-12 08:26
4	max	C536548	POST	ZINC WILLIE WINKIE CANDLE STICK	600	2010-12 17:35

stat.corr() and crosstab()

In [21]:
```
df.stat.corr("Quantity", "UnitPrice")
```

Out[21]:
```
-0.04112314436835551
```

In [23]:
```
df.select(F.corr("Quantity", "UnitPrice").alias("qty_price_corr")).s
```

```
+--------------------+
|      qty_price_corr|
+--------------------+
|-0.04112314436835551|
+--------------------+
```

In [24]:
```
colName = "UnitPrice"
quantileProbs = [0.5]
relError = 0.05
df.stat.approxQuantile("UnitPrice", quantileProbs, relError) # 2.51
```

Out[24]:
```
[2.51]
```

```
In [25]: display(df.stat.crosstab("StockCode", "Quantity").toPandas())
```

	StockCode_Quantity	-1	-10	-12	-2	-24	-3	-4	-5	-6	...	60	600	64	7	70	72	8
0	22578	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	21327	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	22064	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
3	21080	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
4	22219	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1346	47563A	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1347	22224	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1348	46000S	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1349	22680	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1350	22136	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

1351 rows × 67 columns

In [26]:
```
df.stat.freqItems(["StockCode", "Quantity"]).show(truncate=False)
```

```
+----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
--------------------------------------------------------------------+-
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
------------------------------------------------------------+
|StockCode_freqItems
|Quantity_freqItems
|
+----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
--------------------------------------------------------------------+-
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
------------------------------------------------------------+
|[90214E, 20728, 20755, 21703, 22113, 22524, 22041, 72803A, 72798C, 90
181B, 21756, 22694, 90206C, 20970, 21624, 90209C, 84744, 82494L, 2295
2, 20682, 22583, 21705, 20679, 22220, 90177E, 90214A, 22448, 90214S, 2
2121, 22802, 84970L, 72818, 90192, 90200C, 22910, 21380, 90211A, 2113
7, 35271S, 84926A, 20765, 22384, 21524, 22165, 22366, 21221, 21704, 22
519, 85035C, 21967, 22114, 22909, 22900, 22447, 21577, 21877, 20726, 8
5034A, DOT, 84658, 21472, 22804, 22222, 72802C, 21739, 22467, 90214H,
22785, 22446, 22197, 20665, 21733, 22731, 21709, 22086, 40001, 85123A]
|[200, 128, 23, 32, 50, 600, 8, 17, 80, -1, -10, 11, 56, 47, 20, -7,
2, 5, 480, -4, 14, 432, 100, 64, 40, 13, 4, -5, 22, 16, -2, 7, 70, 38
4, 25, 34, 10, 1, 288, 216, 28, 252, 19, 120, 192, 60, 96, 72, 144, 3
6, 27, 9, 18, 48, 21, 12, 3, -6, -24, 30, 15, 33, 6, 24, -12, -3]|
+----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
--------------------------------------------------------------------+-
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
------------------------------------------------------------+
```

monotonically_increasing_id()

monotonically_increasing_id()

In [27]:

```
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 [29]: df = df.withColumn("Id", F.monotonically_increasing_id())

         display(df.toPandas())
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Countr
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Unite Kingdo
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Unite Kingdo
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Unite Kingdo
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Unite Kingdo
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Unite Kingdo
...	...	...	...	...	...	...	...	.
3103	536597	35271S	GOLD PRINT PAPER BAG	14	2010-12-01 17:35:00	0.19	18011.0	Unite Kingdo
3104	536597	21380	WOODEN HAPPY BIRTHDAY GARLAND	1	2010-12-01 17:35:00	2.95	18011.0	Unite Kingdo
3105	536597	22909	SET OF 20 VINTAGE CHRISTMAS NAPKINS	1	2010-12-01 17:35:00	0.85	18011.0	Unite Kingdo
3106	536597	21221	SET/4 BADGES CUTE CREATURES	5	2010-12-01 17:35:00	1.25	18011.0	Unite Kingdo
3107	536597	20755	BLUE PAISLEY POCKET BOOK	6	2010-12-01 17:35:00	0.85	18011.0	Unite Kingdo

3108 rows × 9 columns

initcap, lower, upper

```
In [30]: df.select(F.initcap(F.col("Description"))).show(5, False)  # False
```

```
+-----------------------------------+
|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 [31]: (
df.select(F.col("Description"),
    F.initcap(F.col("Description")),
    F.lower(F.col("Description")),
    F.upper(F.col("Description")))
    .show(2, False)
)

```

```
+-----------------------------------+-----------------------------------
+-----------------------------------+-----------------------------------
+
|Description                        |initcap(Description)
|lower(Description)                 |upper(Description)
|
+-----------------------------------+-----------------------------------
+-----------------------------------+-----------------------------------
+
|WHITE HANGING HEART T-LIGHT HOLDER|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                |WHITE METAL LANTERN
|
+-----------------------------------+-----------------------------------
+-----------------------------------+-----------------------------------
+
only showing top 2 rows
```

ltrim(), rtrim(), trim()

strip spaces (leading, trailing or both)

lpad(), rpad()

pad `char` left or right

In [33]:
```
(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"), 7, " ").alias("lp"),
    F.rpad(F.lit("Hi"), 5, " ").alias("rp"))
    .show(1))
```

```
+---------+---------+-----+-------+-----+
|    ltrim|    rtrim| trim|     lp|   rp|
+---------+---------+-----+-------+-----+
|HELLO    |    HELLO|HELLO|  Hello|Hi   |
+---------+---------+-----+-------+-----+
only showing top 1 row
```

translate()

map char to new one

In [34]:
```
(df.select(F.col("Description"),
           F.translate(F.col("Description"), "LEET", "1337"))
  .show(2,False))
```

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

regexp_replace()

match and replace

regexp_extract()

match and extract

In [35]:
```
regex_string = "BLACK|WHITE|RED|GREEN|BLUE"
(df.select(
    F.col("Description"),
    F.regexp_replace(F.col("Description"), regex_string, "COLOR").al
  .show(2, False))
```

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

In [36]:
```
extract_str = "(BLACK|WHITE|RED|GREEN|BLUE)"
(df.select(
    F.col("Description"),
    F.regexp_extract(F.col("Description"), extract_str, 1).alias("co
  .show(5,False))
```

```
+-----------------------------------+-----------+
|Description                        |color_clean|
+-----------------------------------+-----------+
|WHITE HANGING HEART T-LIGHT HOLDER |WHITE      |
|WHITE METAL LANTERN                |WHITE      |
|CREAM CUPID HEARTS COAT HANGER     |           |
|KNITTED UNION FLAG HOT WATER BOTTLE|           |
|RED WOOLLY HOTTIE WHITE HEART.     |RED        |
+-----------------------------------+-----------+
only showing top 5 rows
```

instr()

find a subsring

In [39]:
```
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", "hasSimpleColor")
   .show(5, False))
```

```
+----------------------------------+--------------+
|Description                       |hasSimpleColor|
+----------------------------------+--------------+
|WHITE HANGING HEART T-LIGHT HOLDER|true          |
|WHITE METAL LANTERN               |true          |
|RED WOOLLY HOTTIE WHITE HEART.    |true          |
|WHITE HANGING HEART T-LIGHT HOLDER|true          |
|WHITE METAL LANTERN               |true          |
+----------------------------------+--------------+
only showing top 5 rows
```

locate() - construct columns dynamically

In [40]:
```
simpleColors = ["black", "white", "red", "green", "blue"]
def color_locator(column, color_string):
  return F.locate(color_string.upper(), column)\
          .cast("boolean")\
          .alias("is_" + color_string)
selectedColumns = [color_locator(df.Description, c) for c in simple(
selectedColumns.append(F.expr("*")) # has to a be Column type
```

In [41]:
```
(df.select(*selectedColumns)
   .where(F.expr("is_white OR is_red"))
   .select("Description","is_white","is_red")
   .show(3, False))
```

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

Datetime

- current_date()
- current_timestamp()

In [42]:
```python
# 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  |2021-04-18|2021-04-18 16:02:08.047|
|1  |2021-04-18|2021-04-18 16:02:08.047|
|2  |2021-04-18|2021-04-18 16:02:08.047|
|3  |2021-04-18|2021-04-18 16:02:08.047|
+---+----------+-----------------------+
only showing top 4 rows
```

to_date(), to_timestamp(), date_add(), date_sub(), datediff(), months_between()

see [additional examples (https://github.com/wgong/py4kids/blob/master/lesson-17-pyspark/spark-guide/notebook/chapter-06-udf_datetime.ipynb)](https://github.com/wgong/py4kids/blob/master/lesson-17-pyspark/spark-guide/notebook/chapter-06-udf_datetime.ipynb) using `udf` to parse datetime

In [52]:
```python
spark.sql("select id, today, date_add(today, -3) as past from dateTa
```

```
+---+----------+----------+
| id|     today|      past|
+---+----------+----------+
|  0|2021-04-18|2021-04-15|
|  1|2021-04-18|2021-04-15|
|  2|2021-04-18|2021-04-15|
+---+----------+----------+
```

In [47]:
```
(
dateDF
    .select("id",
            F.date_sub(F.col("today"), 3).alias("past"),
            "today",
            F.date_add(F.col("today"), 5).alias("future"))
    .show(5)
)
```

```
+---+----------+----------+----------+
| id|      past|     today|    future|
+---+----------+----------+----------+
|  0|2021-04-15|2021-04-18|2021-04-23|
|  1|2021-04-15|2021-04-18|2021-04-23|
|  2|2021-04-15|2021-04-18|2021-04-23|
|  3|2021-04-15|2021-04-18|2021-04-23|
|  4|2021-04-15|2021-04-18|2021-04-23|
+---+----------+----------+----------+
only showing top 5 rows
```

How to work around limitation that 2nd arg of date_add() must be literal `int` value

https://stackoverflow.com/questions/46956026/how-to-convert-column-with-string-type-to-int-form-in-pyspark-data-frame (https://stackoverflow.com/questions/46956026/how-to-convert-column-with-string-type-to-int-form-in-pyspark-data-frame)

In [56]:
```
(
dateDF
    .withColumn("id_days", F.col("id").cast(IntegerType()))
    .withColumn("past", F.expr("date_sub(today, id_days)"))
    .select("id",
            "past",
            "today")
    .show(5)
)
```

```
+---+-------+----------+----------+
| id|id_days|      past|     today|
+---+-------+----------+----------+
|  0|      0|2021-04-18|2021-04-18|
|  1|      1|2021-04-17|2021-04-18|
|  2|      2|2021-04-16|2021-04-18|
|  3|      3|2021-04-15|2021-04-18|
|  4|      4|2021-04-14|2021-04-18|
+---+-------+----------+----------+
only showing top 5 rows
```

In [60]:
```
(
dateDF
    .withColumn("id_days", (F.col("id")+1).cast(IntegerType()))
    .withColumn("past", F.expr("date_sub(today, id_days)"))
    .withColumn("future", F.expr("date_add(today, 2*id_days)"))
    .select("id",
            "past",
            "today",
            "future")
    .show(10)
)
```

```
+---+----------+----------+----------+
| id|      past|     today|    future|
+---+----------+----------+----------+
|  0|2021-04-17|2021-04-18|2021-04-20|
|  1|2021-04-16|2021-04-18|2021-04-22|
|  2|2021-04-15|2021-04-18|2021-04-24|
|  3|2021-04-14|2021-04-18|2021-04-26|
|  4|2021-04-13|2021-04-18|2021-04-28|
|  5|2021-04-12|2021-04-18|2021-04-30|
|  6|2021-04-11|2021-04-18|2021-05-02|
|  7|2021-04-10|2021-04-18|2021-05-04|
|  8|2021-04-09|2021-04-18|2021-05-06|
|  9|2021-04-08|2021-04-18|2021-05-08|
+---+----------+----------+----------+
```

In [61]:
```
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 [62]:
```
dateDF.select(
    F.to_date(F.lit("2016-01-01")).alias("start"),
    F.to_date(F.lit("2017-05-22")).alias("end"))\
    .select("start","end",F.months_between(F.col("start"), F.col("en
    .show(1)
```

```
+----------+----------+------------+
|     start|       end|  month_diff|
+----------+----------+------------+
|2016-01-01|2017-05-22|-16.67741935|
+----------+----------+------------+
only showing top 1 row
```

In [63]:
```
(dateDF
    .withColumn("start", F.to_date(F.lit("2016-01-01")))
    .withColumn("end", F.to_date(F.lit("2017-05-22")))
    .withColumn("month_diff", F.expr("months_between(start, end)"))
    .select("start", "end", "month_diff")
    .show(1)
)
```

```
+----------+----------+------------+
|     start|       end|  month_diff|
+----------+----------+------------+
|2016-01-01|2017-05-22|-16.67741935|
+----------+----------+------------+
only showing top 1 row
```

In [25]:
```
(dateDF
     .withColumn("start", F.to_date(F.lit("2016-01-01")))
     .withColumn("end", F.to_date(F.lit("2017-05-22")))
     .withColumn("month_diff", F.months_between(F.col("start"), F.co
     .select("start", "end", "month_diff")
     .show(1)
)

```

```
+----------+----------+------------+
|     start|       end|  month_diff|
+----------+----------+------------+
|2016-01-01|2017-05-22|-16.67741935|
+----------+----------+------------+
only showing top 1 row
```

reformat date

In [64]:
```
dateFormat = "yyyy-dd-MM"
cleanDateDF = spark.range(1).select(
    F.to_date(F.lit("2017-12-11"), dateFormat).alias("date1"),
    F.to_date(F.lit("2017-20-12"), dateFormat).alias("date2"))
cleanDateDF.createOrReplaceTempView("dateTable2")
cleanDateDF.show()
```

```
+----------+----------+
|     date1|     date2|
+----------+----------+
|2017-11-12|2017-12-20|
+----------+----------+
```

In [65]:
```
spark.sql("select * from dateTable2").show()
```

```
+----------+----------+
|     date1|     date2|
+----------+----------+
|2017-11-12|2017-12-20|
+----------+----------+
```

In [66]:
```
cleanDateDF.select(F.to_timestamp(F.col("date1"), dateFormat))\
    .show()
```

```
+-------------------------------------+
|to_timestamp(`date1`, 'yyyy-dd-MM')|
+-------------------------------------+
|                  2017-11-12 00:00:00|
+-------------------------------------+
```

na.drop(), na.fill(), na.replace()

In [31]:
```
df.na.drop("all", subset=["StockCode", "InvoiceNo"])
```

Out[31]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: string, UnitPrice: double, CustomerID: doub
le, Country: string]
```

In [32]:
```
df.count()
```

Out[32]:
```
3108
```

In [33]:
```
df.na.fill("all", subset=["StockCode", "InvoiceNo"])
```

Out[33]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: string, UnitPrice: double, CustomerID: doub
le, Country: string]
```

In [34]:
```
fill_cols_vals = {"StockCode": 5, "Description" : "No Value"}
df.na.fill(fill_cols_vals)
```

Out[34]:
```
DataFrame[InvoiceNo: string, StockCode: string, Description: string, Q
uantity: int, InvoiceDate: string, UnitPrice: double, CustomerID: doub
le, Country: string]
```

In [35]:
```
df.filter(F.col("Description") == '').show(5,False)
```

```
+---------+---------+-----------+--------+-----------+---------+------
----+-------+
|InvoiceNo|StockCode|Description|Quantity|InvoiceDate|UnitPrice|Custom
erID|Country|
+---------+---------+-----------+--------+-----------+---------+------
----+-------+
+---------+---------+-----------+--------+-----------+---------+------
----+-------+
```

In [67]:
```
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]
```

Complex type

struct()

combine multiple columns into array

In [4]:
```
complexDF = df.select(F.struct("Description", "InvoiceNo").alias("co

complexDF.createOrReplaceTempView("complexDF")
```

In [8]:
```
spark.sql("select * from complexDF").show(5, False)
```

```
+---------------------------------------------+
|complex                                      |
+---------------------------------------------+
|[WHITE HANGING HEART T-LIGHT HOLDER, 536365] |
|[WHITE METAL LANTERN, 536365]                |
|[CREAM CUPID HEARTS COAT HANGER, 536365]     |
|[KNITTED UNION FLAG HOT WATER BOTTLE, 536365]|
|[RED WOOLLY HOTTIE WHITE HEART., 536365]     |
+---------------------------------------------+
only showing top 5 rows
```

split

convert one column into array type

```
In [36]: df.select("Description", F.split(F.col("Description"), " ").alias("d
```

```
+---------------------------------+-----------------------------------
------+
|Description                      |desc_words
|
+---------------------------------+-----------------------------------
------+
|WHITE HANGING HEART T-LIGHT HOLDER|[WHITE, HANGING, HEART, T-LIGHT, H
OLDER]|
|WHITE METAL LANTERN               |[WHITE, METAL, LANTERN]
|
+---------------------------------+-----------------------------------
------+
only showing top 2 rows
```

```
In [41]: df.withColumn("array_col", F.split(F.col("Description"), " "))\
             .selectExpr("Description", "array_col", "array_col[0]","array_co
             .show(5, False)
```

```
+----------------------------------+----------------------------------
---------+-----------+-----------+
|Description                        |array_col
|array_col[0]|array_col[1]|
+----------------------------------+----------------------------------
---------+-----------+-----------+
|WHITE HANGING HEART T-LIGHT HOLDER |[WHITE, HANGING, HEART, T-LIGHT,
HOLDER]  |WHITE       |HANGING     |
|WHITE METAL LANTERN                |[WHITE, METAL, LANTERN]
|WHITE       |METAL       |
|CREAM CUPID HEARTS COAT HANGER     |[CREAM, CUPID, HEARTS, COAT, HANG
ER]      |CREAM       |CUPID       |
|KNITTED UNION FLAG HOT WATER BOTTLE|[KNITTED, UNION, FLAG, HOT, WATE
R, BOTTLE]|KNITTED     |UNION       |
|RED WOOLLY HOTTIE WHITE HEART.     |[RED, WOOLLY, HOTTIE, WHITE, HEAR
T.]      |RED         |WOOLLY      |
+----------------------------------+----------------------------------
---------+-----------+-----------+
only showing top 5 rows
```

size()

In [16]:
```python
df.select("Description",
    F.size(F.split(F.col("Description"), " ")).alias("arr_size"))\
    .show(2, False) # shows 5 and 3
```

```
+-----------------------------------+--------+
|Description                        |arr_size|
+-----------------------------------+--------+
|WHITE HANGING HEART T-LIGHT HOLDER|5       |
|WHITE METAL LANTERN                |3       |
+-----------------------------------+--------+
only showing top 2 rows
```

array_contains

In [42]:
```python
df.select("Description",
        F.array_contains(F.split(F.col("Description"), " "), "WHITE'
    ).show(2,False)
```

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

explode

denorm array column

In [43]:
```python
df.withColumn("splitted", F.split(F.col("Description"), " "))\
  .withColumn("exploded", F.explode(F.col("splitted")))\
  .select("Description", "InvoiceNo", "exploded")\
  .show(10, False)
```

```
+-----------------------------------+---------+--------+
|Description                        |InvoiceNo|exploded|
+-----------------------------------+---------+--------+
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |WHITE   |
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |HANGING |
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |HEART   |
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |T-LIGHT |
|WHITE HANGING HEART T-LIGHT HOLDER|536365   |HOLDER  |
|WHITE METAL LANTERN                |536365   |WHITE   |
|WHITE METAL LANTERN                |536365   |METAL   |
|WHITE METAL LANTERN                |536365   |LANTERN |
|CREAM CUPID HEARTS COAT HANGER     |536365   |CREAM   |
|CREAM CUPID HEARTS COAT HANGER     |536365   |CUPID   |
+-----------------------------------+---------+--------+
only showing top 10 rows
```

map

create a hash map between 2 columns

In [45]:
```
df.select("Description", "InvoiceNo", F.create_map(F.col("Descriptic
  .show(5, False)
```

```
+-----------------------------------+---------+----------------------
------------------------+
|Description                        |InvoiceNo|complex_map
|
+-----------------------------------+---------+----------------------
------------------------+
|WHITE HANGING HEART T-LIGHT HOLDER |536365   |[WHITE HANGING HEART T-
LIGHT HOLDER -> 536365] |
|WHITE METAL LANTERN                |536365   |[WHITE METAL LANTERN ->
536365]                |
|CREAM CUPID HEARTS COAT HANGER     |536365   |[CREAM CUPID HEARTS COA
T HANGER -> 536365]     |
|KNITTED UNION FLAG HOT WATER BOTTLE|536365   |[KNITTED UNION FLAG HOT
WATER BOTTLE -> 536365]|
|RED WOOLLY HOTTIE WHITE HEART.     |536365   |[RED WOOLLY HOTTIE WHIT
E HEART. -> 536365]     |
+-----------------------------------+---------+----------------------
------------------------+
only showing top 5 rows
```

In [46]:
```
df.select(F.create_map(F.col("Description"), F.col("InvoiceNo")).ali
  .selectExpr("complex_map['WHITE METAL LANTERN']")\
    .show(2)
```

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

In [50]:
```
df.withColumn("complex_map", F.create_map(F.col("Description"), F.co
    .selectExpr("Description", "InvoiceNo", "explode(complex_map)")\
    .show(2, False)
```

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

Json

In [52]:
```
jsonDF = spark.range(1).selectExpr("""
  '{"myJSONKey" : {"myJSONValue" : [1, 2, 3]}}' as jsonString""")

```

In [53]:
```
jsonDF.show(2, False)
```

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

In [33]:
```
jsonDF.select(
    F.get_json_object(F.col("jsonString"), "$.myJSONKey.myJSONValue
    F.json_tuple(F.col("jsonString"), "myJSONKey")
    ).show(2, False)
```

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

pack columns into json

In [34]:
```python
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 [35]:
```python
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).alias("old_jso
    .show(2, False)
```

```
+---------------------------------------------+------------------------
-------------------------------------------------+
|old_json                                     |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
```

udf()

In [39]:
```python
udfExampleDF = spark.range(5).toDF("num")
```

In [40]:
```
def power3(double_value):
  return float(double_value ** 3)
power3(2.0)
```

Out[40]:
```
8.0
```

In [41]:
```
power3udf = F.udf(power3)
```

In [42]:
```
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 [43]:
```
spark.udf.register("power3py", power3, DoubleType())
```

Out[43]:
```
<function __main__.power3(double_value)>
```

In [44]:
```
udfExampleDF.selectExpr("power3py(num)").show(5)
# registered via Python
```

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

In [45]:
```
spark.sql("show user functions like 'power*'").show()
```

```
+--------+
|function|
+--------+
|power3py|
+--------+
```

sample question for certification

How to create spark dataframe from list

https://stackoverflow.com/questions/43444925/how-to-create-dataframe-from-list-in-spark-sql/50969995 (https://stackoverflow.com/questions/43444925/how-to-create-dataframe-from-list-in-spark-sql/50969995)

In [ ]:
```
from pyspark.sql.types import *
```

In [79]:
```
test_schema = StructType([
                StructField("Words", StringType())
               ,StructField("Score", IntegerType())
              ])

test_list = [['Hello', 1],
             ['I am fine', 3],
             ['Become Spark Smart', 100]
            ]

test_df = spark.createDataFrame(test_list, schema=test_schema)
test_df.show()
```

```
+------------------+-----+
|             Words|Score|
+------------------+-----+
|             Hello|    1|
|         I am fine|    3|
|Become Spark Smart|  100|
+------------------+-----+
```

Question 1

In [72]:
```
from pyspark.sql import Row
from pyspark.sql.functions import (col,count,desc,sum)

a = [1002, 3001, 4002, 2003, 2002, 3004, 1003, 4006]
# b = spark.createDataFrame(list(map(lambda x: Row(value=x), a)))
```

In [73]:
```
b = (spark
  .createDataFrame(list(map(lambda x: Row(value=x), a)))
  .withColumn("x", F.col("value") % 1000)
)
```

In [74]:
```
b.show()
```

```
+-----+---+
|value|  x|
+-----+---+
| 1002|  2|
| 3001|  1|
| 4002|  2|
| 2003|  3|
| 2002|  2|
| 3004|  4|
| 1003|  3|
| 4006|  6|
+-----+---+
```

In [75]:
```
c = (
    b
    .groupBy(col("x"))
    .agg(count("x"), sum("value"))
    .drop("x")
    .toDF("count", "total")
    .orderBy(col("count").desc(), col("total"))
    .limit(1)
    .show()
)
```

```
+-----+-----+
|count|total|
+-----+-----+
|    3| 7006|
+-----+-----+
```

In [76]:
```
c = b\
    .groupBy(col("x"))\
    .agg(count("x"), sum("value"))\
    .drop("x")\
    .toDF("count", "total")\
    .orderBy(col("count").desc(), col("total"))\
    .limit(1)\
    .show()
```

```
+-----+-----+
|count|total|
+-----+-----+
|    3| 7006|
+-----+-----+
```

In [77]:
```
type(c)
```

Out[77]:
```
NoneType
```

Question 2

In [85]:
```
data_schema = StructType([
                    StructField("UserKey", IntegerType())
                   ,StructField("ItemKey", IntegerType())
                   ,StructField("ItemName", StringType())
                   ,StructField("Score", FloatType())
              ])

data_list = [
  (1, 1000, "Apple", 0.76),
  (2, 1000, "Apple", 0.11),
  (1, 2000, "Orange", 0.98),
  (1, 3000, "Banana", 0.24),
  (2, 3000, "Banana", 0.99)
]

data_df = spark.createDataFrame(data_list, schema=data_schema)
data_df.show()
```

```
+-------+-------+--------+-----+
|UserKey|ItemKey|ItemName|Score|
+-------+-------+--------+-----+
|      1|   1000|   Apple| 0.76|
|      2|   1000|   Apple| 0.11|
|      1|   2000|  Orange| 0.98|
|      1|   3000|  Banana| 0.24|
|      2|   3000|  Banana| 0.99|
+-------+-------+--------+-----+
```

In [84]:
```
(
data_df.groupBy("UserKey")
  .agg(F.sort_array(F.collect_list(F.struct("Score", "ItemKey", "Ite
  .toDF("UserKey", "Collection")
  .show(20, False)
)
```

```
+-------+------------------------------------------------------------
----+
|UserKey|Collection
|
+-------+------------------------------------------------------------
----+
|1      |[[0.98, 2000, Orange], [0.76, 1000, Apple], [0.24, 3000, Bana
na]]|
|2      |[[0.99, 3000, Banana], [0.11, 1000, Apple]]
|
+-------+------------------------------------------------------------
----+
```

Question 3 - windowSpec

In [105]:

```
people_schema = StructType([
                  StructField("name", StringType())
                 ,StructField("department", IntegerType())
                 ,StructField("score", ArrayType(IntegerType()))
              ])

people_list = [
    ("Ali", 0, [100]),
    ("Barbara", 1, [300, 250, 100]),
    ("Cesar", 1, [350, 100]),
    ("Dongmei", 1, [400, 100]),
    ("Eli", 2, [250]),
    ("Florita", 2, [500, 300, 100]),
    ("Gatimu", 3, [300, 100])
]


people_df = spark.createDataFrame(people_list, schema=people_schema)
people_df.show()
```

```
+-------+----------+---------------+
|   name|department|          score|
+-------+----------+---------------+
|    Ali|         0|          [100]|
|Barbara|         1|[300, 250, 100]|
|  Cesar|         1|     [350, 100]|
|Dongmei|         1|     [400, 100]|
|    Eli|         2|          [250]|
|Florita|         2|[500, 300, 100]|
| Gatimu|         3|     [300, 100]|
+-------+----------+---------------+
```

In [ ]:

```
from pyspark.sql.window import Window
from pyspark.sql.functions import explode, dense_rank, max

windowSpec = Window.partitionBy("department").orderBy(F.col("score")
```

In [109]:

```
# look at intermediate result
(
people_df
  .withColumn("score", explode(col("score")))
  .select(
    col("department"),
    col("name"),
    col("score"),
    dense_rank().over(windowSpec).alias("rank"),
    max(col("score")).over(windowSpec).alias("highest")
  )
  .show()
)
```

```
+----------+-------+-----+----+-------+
|department|   name|score|rank|highest|
+----------+-------+-----+----+-------+
|         1|Dongmei|  400|   1|    400|
|         1|  Cesar|  350|   2|    400|
|         1|Barbara|  300|   3|    400|
|         1|Barbara|  250|   4|    400|
|         1|Barbara|  100|   5|    400|
|         1|  Cesar|  100|   5|    400|
|         1|Dongmei|  100|   5|    400|
|         3| Gatimu|  300|   1|    300|
|         3| Gatimu|  100|   2|    300|
|         2|Florita|  500|   1|    500|
|         2|Florita|  300|   2|    500|
|         2|    Eli|  250|   3|    500|
|         2|Florita|  100|   4|    500|
|         0|    Ali|  100|   1|    100|
+----------+-------+-----+----+-------+
```

In [110]:

```
(
people_df
  .withColumn("score", explode(col("score")))
  .select(
    col("department"),
    col("name"),
    dense_rank().over(windowSpec).alias("rank"),
    max(col("score")).over(windowSpec).alias("highest")
  )
  .where(col("rank") == 1)
  .drop("rank")
  .orderBy("department")
  .show()
)
```

```
+----------+-------+-------+
|department|   name|highest|
+----------+-------+-------+
|         0|    Ali|    100|
|         1|Dongmei|    400|
|         2|Florita|    500|
|         3| Gatimu|    300|
+----------+-------+-------+
```

In [ ]:

```
```