HW2-solution

March 29, 2025

[1]:
```
# new way
import pyspark

spark = pyspark.sql.SparkSession.builder.config('spark.driver.memory','8g').
 ↪getOrCreate()
spark
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
25/03/29 09:25:56 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

[1]:
```
<pyspark.sql.session.SparkSession at 0x7a067b442ab0>
```

1 Q1 - Bakery items sold by day

[54]:
```
bakery = spark.read\
    .option("header", True)\
    .option("inferSchema", True)\
    .csv("shared/hw2/Bakery.csv")
bakery.limit(5).toPandas()
```

[54]:

	Date	Time	Transaction	Item
0	2016-10-30	2025-03-29 09:58:11	1	Bread
1	2016-10-30	2025-03-29 10:05:34	2	Scandinavian
2	2016-10-30	2025-03-29 10:05:34	2	Scandinavian
3	2016-10-30	2025-03-29 10:07:57	3	Hot chocolate
4	2016-10-30	2025-03-29 10:07:57	3	Jam

[55]:
```
bakery.printSchema()
```

```
root
 |-- Date: date (nullable = true)
 |-- Time: timestamp (nullable = true)
 |-- Transaction: integer (nullable = true)
```

```
 |-- Item: string (nullable = true)
```

[59]:
```
from pyspark.sql.functions import dayofweek, hour, col
bakery\
    .withColumn("day", dayofweek("Date"))\
    .withColumn("hour", hour("Time"))\
    .show(5)
```

```
+----------+-------------------+-----------+-------------+---+----+
|      Date|               Time|Transaction|         Item|day|hour|
+----------+-------------------+-----------+-------------+---+----+
|2016-10-30|2025-03-29 09:58:11|          1|        Bread|  1|   9|
|2016-10-30|2025-03-29 10:05:34|          2| Scandinavian|  1|  10|
|2016-10-30|2025-03-29 10:05:34|          2| Scandinavian|  1|  10|
|2016-10-30|2025-03-29 10:07:57|          3|Hot chocolate|  1|  10|
|2016-10-30|2025-03-29 10:07:57|          3|          Jam|  1|  10|
+----------+-------------------+-----------+-------------+---+----+
only showing top 5 rows
```

[75]:
```
from pyspark.sql.functions import lit, expr

# use a SQL expression
sqlexp = '''CASE
    WHEN daynum = 1 THEN 'Sunday'
    WHEN daynum = 2 THEN 'Monday'
    WHEN daynum = 3 THEN 'Tuesday'
    WHEN daynum = 4 THEN 'Wednesday'
    WHEN daynum = 5 THEN 'Thursday'
    WHEN daynum = 6 THEN 'Friday'
    WHEN daynum = 7 THEN 'Saturday'
END
'''

q1 = bakery\
    .withColumn("daynum", dayofweek("Date"))\
    .withColumn("hour", hour("Time"))\
    .drop("Date", "Time", "Transaction")\
    .withColumn("day", expr(sqlexp))\
    .select("Item", "day")\
    .groupBy("Item", "day").count()\
    .orderBy("Day", "Item")
q1.limit(5).toPandas()
```

[75]:

	Item	day	count
0	Afternoon with the baker	Friday	7
1	Alfajores	Friday	59
2	Art Tray	Friday	4
3	Baguette	Friday	21
4	Bakewell	Friday	3

2 Q2 - most popular bakery items by day type

[79]:
```
sqlexp = '''CASE
    WHEN day = 'Saturday' THEN 'Weekend'
    WHEN day = 'Sunday' THEN 'Weekend'
    ELSE 'Weekday'
END
'''

q2 = q1.select(col('*'), expr(sqlexp).alias('daytype'))
q2.limit(5).toPandas()
```

[79]:

	Item	day	count	daytype
0	Afternoon with the baker	Friday	7	Weekday
1	Alfajores	Friday	59	Weekday
2	Art Tray	Friday	4	Weekday
3	Baguette	Friday	21	Weekday
4	Bakewell	Friday	3	Weekday

[88]:
```
from pyspark.sql.window import Window
from pyspark.sql.functions import col, row_number, desc, asc, sum

windowTop = Window.partitionBy("daytype").orderBy(col("count").desc())
top = q2\
    .groupBy("daytype", "Item").sum()\
    .withColumnRenamed("sum(count)", "count")\
    .withColumn("row",row_number().over(windowTop)) \
    .filter(col("row") <= 2)\
    .groupBy("daytype").agg(collect_set("Item").alias("Top Items"))

top.limit(5).toPandas()
```

[88]:

	daytype	Top Items
0	Weekday	[Coffee, Bread]
1	Weekend	[Coffee, Bread]

3 Q3 - Population

[2]:
```
pop = spark.read\
    .option("header", True)\
    .option("inferSchema", True)\
    .csv("shared/hw2/populationbycountry19802010millions.csv")
pop.limit(5).toPandas()
```

```
25/03/29 09:26:16 WARN SparkStringUtils: Truncated the string representation of
a plan since it was too large. This behavior can be adjusted by setting
'spark.sql.debug.maxToStringFields'.
25/03/29 09:26:17 WARN CSVHeaderChecker: CSV header does not conform to the
schema.
 Header: , 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990,
1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
2004, 2005, 2006, 2007, 2008, 2009, 2010
 Schema: _c0, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990,
1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
2004, 2005, 2006, 2007, 2008, 2009, 2010
Expected: _c0 but found:
CSV file: file:///home/jovyan/shared/hw2/populationbycountry19802010millions.csv
```

[2]:

	_c0	1980	1981	1982	1983	1984	\
0	North America	320.27638	324.44694	328.62014	332.72487	336.72143	
1	Bermuda	0.05473	0.05491	0.05517	0.05551	0.05585	
2	Canada	24.5933	24.9	25.2019	25.4563	25.7018	
3	Greenland	0.05021	0.05103	0.05166	0.05211	0.05263	
4	Mexico	68.34748	69.96926	71.6409	73.36288	75.08014	

	1985	1986	1987	1988	…	2001	2002	\
0	340.74811	344.89548	349.07829	353.2939	…	417.83236	422.05268	
1	0.05618	0.05651	0.05683	0.05717	…	0.06361	0.06418	
2	25.9416	26.2038	26.5497	26.8948	…	31.37674	31.64096	
3	0.05315	0.05364	0.0541	0.05485	…	0.05713	0.05736	
4	76.76723	78.44243	80.12249	81.78182	…	101.24696	102.47993	

	2003	2004	2005	2006	2007	2008	\
0	426.06238	430.26938	434.47232	438.82964	443.3473	447.67394	
1	0.06476	0.06534	0.06591	0.06644	0.06692	0.06739	
2	31.88931	32.13476	32.38638	32.65668	32.93596	33.2127	
3	0.05754	0.0577	0.05778	0.05764	0.05753	0.05756	
4	103.71806	104.95959	106.2029	107.44953	108.70089	109.9554	

	2009	2010
0	451.83698	456.59331
1	0.06784	0.06827
2	33.48721	33.75974
3	0.0576	0.05764

```
 4  111.21179  112.46886

 [5 rows x 32 columns]
```

[3]:
```
pop.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- 1980: string (nullable = true)
 |-- 1981: string (nullable = true)
 |-- 1982: string (nullable = true)
 |-- 1983: string (nullable = true)
 |-- 1984: string (nullable = true)
 |-- 1985: string (nullable = true)
 |-- 1986: string (nullable = true)
 |-- 1987: string (nullable = true)
 |-- 1988: string (nullable = true)
 |-- 1989: string (nullable = true)
 |-- 1990: string (nullable = true)
 |-- 1991: string (nullable = true)
 |-- 1992: string (nullable = true)
 |-- 1993: string (nullable = true)
 |-- 1994: string (nullable = true)
 |-- 1995: string (nullable = true)
 |-- 1996: string (nullable = true)
 |-- 1997: string (nullable = true)
 |-- 1998: string (nullable = true)
 |-- 1999: string (nullable = true)
 |-- 2000: string (nullable = true)
 |-- 2001: string (nullable = true)
 |-- 2002: string (nullable = true)
 |-- 2003: string (nullable = true)
 |-- 2004: string (nullable = true)
 |-- 2005: string (nullable = true)
 |-- 2006: string (nullable = true)
 |-- 2007: string (nullable = true)
 |-- 2008: string (nullable = true)
 |-- 2009: string (nullable = true)
 |-- 2010: string (nullable = true)
```

[4]:
```
# cache pop to compute faster and clear warnings
pop.cache()
pop.count()
```

```
25/03/29 09:26:18 WARN CSVHeaderChecker: CSV header does not conform to the
schema.
 Header: , 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990,
```

```
1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
2004, 2005, 2006, 2007, 2008, 2009, 2010
 Schema: _c0, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990,
1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
2004, 2005, 2006, 2007, 2008, 2009, 2010
Expected: _c0 but found:
CSV file: file:///home/jovyan/shared/hw2/populationbycountry19802010millions.csv
```

[4]:
```
232
```

[6]:
```
# convert to double
from pyspark.sql.functions import col, cast
from pyspark.sql.types import DoubleType

for yr in range(1980, 2011):
    curr = str(yr)
    pop = pop.withColumn(curr, col(curr).cast(DoubleType()))
```

[7]:
```
# clean up the countries
pop = pop.withColumnRenamed("_c0", "country")
countries = [c["country"] for c in pop.select("country").collect()]
print(countries)
```

```
['North America', 'Bermuda', 'Canada', 'Greenland', 'Mexico', 'Saint Pierre and
Miquelon', 'United States', 'Central & South America', 'Antarctica', 'Antigua
and Barbuda', 'Argentina', 'Aruba', 'Bahamas, The', 'Barbados', 'Belize',
'Bolivia', 'Brazil', 'Cayman Islands', 'Chile', 'Colombia', 'Costa Rica',
'Cuba', 'Dominica', 'Dominican Republic', 'Ecuador', 'El Salvador', 'Falkland
Islands (Islas Malvinas)', 'French Guiana', 'Grenada', 'Guadeloupe',
'Guatemala', 'Guyana', 'Haiti', 'Honduras', 'Jamaica', 'Martinique',
'Montserrat', 'Netherlands Antilles', 'Nicaragua', 'Panama', 'Paraguay', 'Peru',
'Puerto Rico', 'Saint Kitts and Nevis', 'Saint Lucia', 'Saint
Vincent/Grenadines', 'Suriname', 'Trinidad and Tobago', 'Turks and Caicos
Islands', 'Uruguay', 'Venezuela', 'Virgin Islands,  U.S.', 'Virgin Islands,
British', 'Europe', 'Albania', 'Austria', 'Belgium', 'Bosnia and Herzegovina',
'Bulgaria', 'Croatia', 'Cyprus', 'Czech Republic', 'Denmark', 'Faroe Islands',
'Finland', 'Former Czechoslovakia', 'Former Serbia and Montenegro', 'Former
Yugoslavia', 'France', 'Germany', 'Germany, East', 'Germany, West', 'Gibraltar',
'Greece', 'Hungary', 'Iceland', 'Ireland', 'Italy', 'Luxembourg', 'Macedonia',
'Malta', 'Montenegro', 'Netherlands', 'Norway', 'Poland', 'Portugal', 'Romania',
'Serbia', 'Slovakia', 'Slovenia', 'Spain', 'Sweden', 'Switzerland', 'Turkey',
'United Kingdom', 'Eurasia', 'Armenia', 'Azerbaijan', 'Belarus', 'Estonia',
'Former U.S.S.R.', 'Georgia', 'Kazakhstan', 'Kyrgyzstan', 'Latvia', 'Lithuania',
'Moldova', 'Russia', 'Tajikistan', 'Turkmenistan', 'Ukraine', 'Uzbekistan',
'Middle East', 'Bahrain', 'Iran', 'Iraq', 'Israel', 'Jordan', 'Kuwait',
'Lebanon', 'Oman', 'Palestine', 'Qatar', 'Saudi Arabia', 'Syria', 'United Arab
Emirates', 'Yemen', 'Africa', 'Algeria', 'Angola', 'Benin', 'Botswana', 'Burkina
Faso', 'Burundi', 'Cameroon', 'Cape Verde', 'Central African Republic', 'Chad',
```

```
'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Cote dIvoire
(IvoryCoast)', 'Djibouti', 'Egypt', 'Equatorial Guinea', 'Eritrea', 'Ethiopia',
'Gabon', 'Gambia, The', 'Ghana', 'Guinea', 'Guinea-Bissau', 'Kenya', 'Lesotho',
'Liberia', 'Libya', 'Madagascar', 'Malawi', 'Mali', 'Mauritania', 'Mauritius',
'Morocco', 'Mozambique', 'Namibia', 'Niger', 'Nigeria', 'Reunion', 'Rwanda',
'Saint Helena', 'Sao Tome and Principe', 'Senegal', 'Seychelles', 'Sierra
Leone', 'Somalia', 'South Africa', 'Sudan', 'Swaziland', 'Tanzania', 'Togo',
'Tunisia', 'Uganda', 'Western Sahara', 'Zambia', 'Zimbabwe', 'Asia & Oceania',
'Afghanistan', 'American Samoa', 'Australia', 'Bangladesh', 'Bhutan', 'Brunei',
'Burma (Myanmar)', 'Cambodia', 'China', 'Cook Islands', 'Fiji', 'French
Polynesia', 'Guam', 'Hawaiian Trade Zone', 'Hong Kong', 'India', 'Indonesia',
'Japan', 'Kiribati', 'Korea, North', 'Korea, South', 'Laos', 'Macau',
'Malaysia', 'Maldives', 'Mongolia', 'Nauru', 'Nepal', 'New Caledonia', 'New
Zealand', 'Niue', 'Pakistan', 'Papua New Guinea', 'Philippines', 'Samoa',
'Singapore', 'Solomon Islands', 'Sri Lanka', 'Taiwan', 'Thailand', 'Timor-Leste
(East Timor)', 'Tonga', 'U.S. Pacific Islands', 'Vanuatu', 'Vietnam', 'Wake
Island', 'World']
```

[8]:
```
regions = ['North America', 'Central & South America', 'World']

#filter these out
from pyspark.sql.functions import array_remove, col
for r in regions:
    pop = pop.filter(col("country") != r)

# check
print( [c["country"] for c in pop.select("country").collect()])
```

```
['Bermuda', 'Canada', 'Greenland', 'Mexico', 'Saint Pierre and Miquelon',
'United States', 'Antarctica', 'Antigua and Barbuda', 'Argentina', 'Aruba',
'Bahamas, The', 'Barbados', 'Belize', 'Bolivia', 'Brazil', 'Cayman Islands',
'Chile', 'Colombia', 'Costa Rica', 'Cuba', 'Dominica', 'Dominican Republic',
'Ecuador', 'El Salvador', 'Falkland Islands (Islas Malvinas)', 'French Guiana',
'Grenada', 'Guadeloupe', 'Guatemala', 'Guyana', 'Haiti', 'Honduras', 'Jamaica',
'Martinique', 'Montserrat', 'Netherlands Antilles', 'Nicaragua', 'Panama',
'Paraguay', 'Peru', 'Puerto Rico', 'Saint Kitts and Nevis', 'Saint Lucia',
'Saint Vincent/Grenadines', 'Suriname', 'Trinidad and Tobago', 'Turks and Caicos
Islands', 'Uruguay', 'Venezuela', 'Virgin Islands,  U.S.', 'Virgin Islands,
British', 'Europe', 'Albania', 'Austria', 'Belgium', 'Bosnia and Herzegovina',
'Bulgaria', 'Croatia', 'Cyprus', 'Czech Republic', 'Denmark', 'Faroe Islands',
'Finland', 'Former Czechoslovakia', 'Former Serbia and Montenegro', 'Former
Yugoslavia', 'France', 'Germany', 'Germany, East', 'Germany, West', 'Gibraltar',
'Greece', 'Hungary', 'Iceland', 'Ireland', 'Italy', 'Luxembourg', 'Macedonia',
'Malta', 'Montenegro', 'Netherlands', 'Norway', 'Poland', 'Portugal', 'Romania',
'Serbia', 'Slovakia', 'Slovenia', 'Spain', 'Sweden', 'Switzerland', 'Turkey',
'United Kingdom', 'Eurasia', 'Armenia', 'Azerbaijan', 'Belarus', 'Estonia',
'Former U.S.S.R.', 'Georgia', 'Kazakhstan', 'Kyrgyzstan', 'Latvia', 'Lithuania',
'Moldova', 'Russia', 'Tajikistan', 'Turkmenistan', 'Ukraine', 'Uzbekistan',
```

```
'Middle East', 'Bahrain', 'Iran', 'Iraq', 'Israel', 'Jordan', 'Kuwait',
'Lebanon', 'Oman', 'Palestine', 'Qatar', 'Saudi Arabia', 'Syria', 'United Arab
Emirates', 'Yemen', 'Africa', 'Algeria', 'Angola', 'Benin', 'Botswana', 'Burkina
Faso', 'Burundi', 'Cameroon', 'Cape Verde', 'Central African Republic', 'Chad',
'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Cote dIvoire
(IvoryCoast)', 'Djibouti', 'Egypt', 'Equatorial Guinea', 'Eritrea', 'Ethiopia',
'Gabon', 'Gambia, The', 'Ghana', 'Guinea', 'Guinea-Bissau', 'Kenya', 'Lesotho',
'Liberia', 'Libya', 'Madagascar', 'Malawi', 'Mali', 'Mauritania', 'Mauritius',
'Morocco', 'Mozambique', 'Namibia', 'Niger', 'Nigeria', 'Reunion', 'Rwanda',
'Saint Helena', 'Sao Tome and Principe', 'Senegal', 'Seychelles', 'Sierra
Leone', 'Somalia', 'South Africa', 'Sudan', 'Swaziland', 'Tanzania', 'Togo',
'Tunisia', 'Uganda', 'Western Sahara', 'Zambia', 'Zimbabwe', 'Asia & Oceania',
'Afghanistan', 'American Samoa', 'Australia', 'Bangladesh', 'Bhutan', 'Brunei',
'Burma (Myanmar)', 'Cambodia', 'China', 'Cook Islands', 'Fiji', 'French
Polynesia', 'Guam', 'Hawaiian Trade Zone', 'Hong Kong', 'India', 'Indonesia',
'Japan', 'Kiribati', 'Korea, North', 'Korea, South', 'Laos', 'Macau',
'Malaysia', 'Maldives', 'Mongolia', 'Nauru', 'Nepal', 'New Caledonia', 'New
Zealand', 'Niue', 'Pakistan', 'Papua New Guinea', 'Philippines', 'Samoa',
'Singapore', 'Solomon Islands', 'Sri Lanka', 'Taiwan', 'Thailand', 'Timor-Leste
(East Timor)', 'Tonga', 'U.S. Pacific Islands', 'Vanuatu', 'Vietnam', 'Wake
Island']
```

[9]:
```
# compute change; drop any rows with missing values
for yr in range(1981, 2011):
    curr = str(yr)
    prev = str(yr-1)
    pop = pop\
        .withColumn("_"+curr, 100.0 * ((col(curr) - col(prev))/col(prev)))\
        .drop(col(prev))
pop = pop.drop("2010")
pop.limit(5).toPandas()
```

[9]:

	country	_1981	_1982	_1983	_1984	\
0	Bermuda	0.328887	0.473502	0.616277	0.612502	
1	Canada	1.247088	1.212450	1.009448	0.964398	
2	Greenland	1.633141	1.234568	0.871080	0.997889	
3	Mexico	2.372845	2.389106	2.403627	2.340775	
4	Saint Pierre and Miquelon	0.333890	0.665557	0.330579	0.658979	

	_1985	_1986	_1987	_1988	_1989	…	_2001	_2002	\
0	0.590868	0.587398	0.566271	0.598276	0.559734	…	0.872185	0.896086	
1	0.933009	1.010732	1.320038	1.299826	1.801463	…	0.891267	0.842089	
2	0.988030	0.921919	0.857569	1.386322	1.020966	…	0.421867	0.402591	
3	2.247052	2.182181	2.141775	2.070992	1.938108	…	1.321310	1.217785	
4	0.818331	0.811688	0.644122	0.480000	0.477707	…	-0.624025	-0.627943	

	_2003	_2004	_2005	_2006	_2007	_2008	_2009	\

```
0  0.903708  0.895615  0.872360  0.804127  0.722456  0.702331  0.667755
1  0.784900  0.769694  0.783015  0.834610  0.855200  0.840237  0.826521
2  0.313808  0.278067  0.138648 -0.242298 -0.190840  0.052147  0.069493
3  1.208168  1.197024  1.184561  1.173819  1.164603  1.154094  1.142636
4 -0.631912 -0.635930 -0.800000 -0.806452 -0.813008 -0.819672 -0.826446

      _2010
0  0.633844
1  0.813833
2  0.069444
3  1.130339
4 -1.000000

[5 rows x 31 columns]
```

[10]:
```
# get first year out
from pyspark.sql.functions import lit

q3 = pop.select("country", lit('1981').alias("year"), col('_1981').
 ↪alias("change"))
q3.limit(5).toPandas()
```

[10]:
```
                     country  year    change
0                    Bermuda  1981  0.328887
1                     Canada  1981  1.247088
2                  Greenland  1981  1.633141
3                     Mexico  1981  2.372845
4  Saint Pierre and Miquelon  1981  0.333890
```

[11]:
```
# pivot all other years
from pyspark.sql.functions import desc
for yr in range(1982, 2011):
    curr = str(yr)
    c = pop.select("country", lit(curr).alias("year"), col('_'+curr).
 ↪alias("change"))
    q3 = q3.union(c)
q3.limit(5).toPandas()
```

[11]:
```
                     country  year    change
0                    Bermuda  1981  0.328887
1                     Canada  1981  1.247088
2                  Greenland  1981  1.633141
3                     Mexico  1981  2.372845
4  Saint Pierre and Miquelon  1981  0.333890
```

[33]:
```
# drop non numeric
from pyspark.sql.functions import isnull
```

```
q3 = q3.filter(isnull("change") == False)
```

[34]:
```
q3.orderBy("change").limit(5).toPandas()
```

[34]:

	country	year	change
0	Kuwait	1991	-55.453162
1	Montserrat	1998	-43.193277
2	Montserrat	1997	-25.157233
3	Netherlands Antilles	1986	-24.587817
4	Montserrat	1996	-22.590068

[40]:
```
# per year, find the biggest and smallest
from pyspark.sql.window import Window
from pyspark.sql.functions import col, row_number, desc, asc

windowTop = Window.partitionBy("year").orderBy(col("change").desc())
top = q3.withColumn("row",row_number().over(windowTop)) \
  .filter(col("row") < 2)\

windowBottom = Window.partitionBy("year").orderBy(col("change").asc())
bottom = q3.withColumn("row",row_number().over(windowBottom)) \
  .filter(col("row") < 2)\

df = top.union(bottom).orderBy([asc("year"), desc("change")])
df.toPandas()
```

[40]:

	country	year	change	row
0	Western Sahara	1981	12.133183	1
1	Afghanistan	1981	-9.106331	1
2	Western Sahara	1982	11.115105	1
3	Afghanistan	1982	-8.017227	1
4	French Guiana	1983	14.285714	1
5	Antigua and Barbuda	1983	-3.514189	1
6	Qatar	1984	10.964057	1
7	Antigua and Barbuda	1984	-1.752514	1
8	French Guiana	1985	12.500000	1
9	Cook Islands	1985	-1.409245	1
10	Qatar	1986	8.771733	1
11	Netherlands Antilles	1986	-24.587817	1
12	French Guiana	1987	11.111111	1
13	Saint Helena	1987	-21.299639	1
14	Cayman Islands	1988	11.010421	1
15	Mozambique	1988	-2.883632	1
16	United Arab Emirates	1989	6.119858	1

17	Somalia	1989	-2.196497	1
18	Djibouti	1990	12.824048	1
19	Liberia	1990	-12.816300	1
20	Jordan	1991	11.273940	1
21	Kuwait	1991	-55.453162	1
22	Kuwait	1992	48.633439	1
23	Somalia	1992	-5.387440	1
24	Afghanistan	1993	13.224595	1
25	Bosnia and Herzegovina	1993	-7.072117	1
26	Afghanistan	1994	8.727662	1
27	Rwanda	1994	-14.363511	1
28	Burundi	1995	7.222489	1
29	Rwanda	1995	-15.871881	1
30	Rwanda	1996	19.614177	1
31	Montserrat	1996	-22.590068	1
32	Falkland Islands (Islas Malvinas)	1997	21.500000	1
33	Montserrat	1997	-25.157233	1
34	Liberia	1998	12.017450	1
35	Montserrat	1998	-43.193277	1
36	Falkland Islands (Islas Malvinas)	1999	7.692308	1
37	Cook Islands	1999	-2.991945	1
38	Montserrat	2000	16.863905	1
39	Cook Islands	2000	-3.262159	1
40	Montserrat	2001	7.341772	1
41	Cook Islands	2001	-3.556101	1
42	Montserrat	2002	13.443396	1
43	Cook Islands	2002	-3.687222	1
44	Afghanistan	2003	5.803892	1
45	Montserrat	2003	-6.652807	1
46	Montserrat	2004	10.467706	1
47	Djibouti	2004	-4.830771	1
48	Liberia	2005	4.797671	1
49	Montserrat	2005	-8.669355	1
50	Jordan	2006	7.088497	1
51	Nauru	2006	-4.395604	1
52	Jordan	2007	6.764378	1
53	Nauru	2007	-4.702194	1
54	Montserrat	2008	12.638581	1
55	Cook Islands	2008	-3.309693	1
56	Liberia	2009	4.157111	1
57	Cook Islands	2009	-3.259984	1
58	Niger	2010	3.737166	1
59	Cook Islands	2010	-3.201348	1

4 Q4 - Word Count

[41]:
```
q4 = spark.read.text("shared/hw2/hw1dir/*.txt")
q4.limit(5).toPandas()
```

[41]:
```
                                                value
0
1  @@31754341 <h> COVID-19 : 73 Children Grabbed …
2  @@31754441 <p> ECONOMYNEXT- Sri Lanka has rela…
3  @@31754641 <p> The Colombo Stock Exchange ( CS…
4  @@31754741 <h> Over 80% of US now in lockdown …
```

[42]:
```
# cleanup text and normalize
from pyspark.sql.functions import regexp_replace, lower, trim, col
df = q4.select(trim(regexp_replace(lower("value"), "[^0-9a-z]", " ")).
 ↪alias("text"))\
    .select(regexp_replace("text", " +", " ").alias("text"))\
    .select(regexp_replace("text", "^ +$", "").alias("text"))\
    .filter(col("text") != '')\
    .na.drop()
df.limit(5).toPandas()
```

[42]:
```
                                                text
0  31754341 h covid 19 73 children grabbed in tak…
1  31754441 p economynext sri lanka has relaxed a…
2  31754641 p the colombo stock exchange cse has …
3  31754741 h over 80 of us now in lockdown p fou…
4  31755241 h the virus is a reminder of somethin…
```

[43]:
```
# tokenize
from pyspark.ml.feature import Tokenizer, NGram
tokenizer = Tokenizer()
tokenizer.setInputCol("text")
tokenizer.setOutputCol("words")
df = tokenizer.transform(df)
df.limit(5).toPandas()
```

[43]:
```
                                                text  \
0  31754341 h covid 19 73 children grabbed in tak…
1  31754441 p economynext sri lanka has relaxed a…
2  31754641 p the colombo stock exchange cse has …
3  31754741 h over 80 of us now in lockdown p fou…
4  31755241 h the virus is a reminder of somethin…

                                               words
0  [31754341, h, covid, 19, 73, children, grabbed…
1  [31754441, p, economynext, sri, lanka, has, re…
```

```
2  [31754641, p, the, colombo, stock, exchange, c…
3  [31754741, h, over, 80, of, us, now, in, lockd…
4  [31755241, h, the, virus, is, a, reminder, of,…
```

[44]:
```
# explode (pivot)
from pyspark.sql.functions import explode
df = df.select(explode("words").alias("word"))
df.limit(5).toPandas()
```

[44]:
```
       word
0  31754341
1         h
2     covid
3        19
4        73
```

[45]:
```
# count, sort, pick to 20
from pyspark.sql.functions import desc
df.groupBy("word").count()\
    .orderBy(desc("count"))\
    .show()
```

```
[Stage 78:==============================>                          (1 + 1) / 2]

+----+------+
|word| count|
+----+------+
| the|163547|
|  to| 89046|
|   p| 78664|
|  of| 75568|
| and| 72730|
|  in| 56782|
|   a| 53198|
| for| 29770|
|that| 28852|
|  is| 27601|
|  on| 24485|
|   s| 23615|
|with| 19575|
| are| 19417|
|  it| 18231|
|  be| 17998|
|  as| 17796|
|have| 16188|
|  at| 15965|
|  we| 15754|
+----+------+
```

only showing top 20 rows

5 Extra Credit - Trigrams

[46]:
```
# there are no newlines in the input, which is the default separator for text␣
 ↪files.
# we'll pick a new separator, say lines , i.e. '.'
extra_credit = spark.read.option("lineSep", ".").text("shared/hw2/
 ↪internet_archive_scifi_v3.txt")
extra_credit
```

[46]: DataFrame[value: string]

[47]:
```
# verify we have a suitable dataframe
print(extra_credit.count())
extra_credit.limit(5).toPandas()
```

[Stage 81:> (0 + 2) / 2]

2123243

[47]:

	value
0	MARCH # All Stories New and Complete Publisher…
1	, Kingston, New York
2	Volume #, No
3	#
4	Copyright # by Quinn Publishing Company, Inc

[48]:
```
# do text cleanup : *** THIS WAS NOT REQUIRED ****
from pyspark.sql.functions import regexp_replace, lower, trim, col
df = extra_credit.select(trim(regexp_replace(lower("value"), "[^0-9a-z]", " ")).
 ↪alias("text"))\
    .select(regexp_replace("text", " +", " ").alias("text"))\
    .select(regexp_replace("text", "^ +$", "").alias("text"))\
    .filter(col("text") != '')\
    .na.drop()
df.limit(5).toPandas()
```

[48]:

	text
0	march all stories new and complete publisher e…
1	kingston new york
2	volume no
3	copyright by quinn publishing company inc
4	application for entry as second class matter a…

[49]:
```
# tokenize
from pyspark.ml.feature import Tokenizer, NGram
tokenizer = Tokenizer()
tokenizer.setInputCol("text")
tokenizer.setOutputCol("words")
df = tokenizer.transform(df)
df.limit(5).toPandas()
```

[49]:
```
                                                text  \
0  march all stories new and complete publisher e…
1                                  kingston new york
2                                         volume no
3          copyright by quinn publishing company inc
4  application for entry as second class matter a…

                                               words
0  [march, all, stories, new, and, complete, publ…
1                              [kingston, new, york]
2                                   [volume, no]
3   [copyright, by, quinn, publishing, company, inc]
4  [application, for, entry, as, second, class, m…
```

[50]:
```
# filter out lines with less than 3 words
from pyspark.sql.functions import array_size
df = df.select("words").filter(array_size("words")>=3)
df.limit(5).toPandas()
```

[50]:
```
                                               words
0  [march, all, stories, new, and, complete, publ…
1                              [kingston, new, york]
2   [copyright, by, quinn, publishing, company, inc]
3  [application, for, entry, as, second, class, m…
4                [subscription, for, issues, in, u]
```

[51]:
```
# make trigrams
from pyspark.ml.feature import NGram
trigram = NGram(inputCol="words", outputCol="trigrams", n=3)
df = trigram.transform(df)
df.limit(5).toPandas()
```

[51]:
```
                                               words  \
0  [march, all, stories, new, and, complete, publ…
1                              [kingston, new, york]
2   [copyright, by, quinn, publishing, company, inc]
3  [application, for, entry, as, second, class, m…
4                [subscription, for, issues, in, u]
```

```
                                                 trigrams
0  [march all stories, all stories new, stories n…
1                                [kingston new york]
2  [copyright by quinn, by quinn publishing, quin…
3  [application for entry, for entry as, entry as…
4  [subscription for issues, for issues in, issue…
```

[52]:
```
# explode the trigrams (pivot)
from pyspark.sql.functions import explode
df = df.select(explode("trigrams").alias("trigram"))
df.limit(10).toPandas()
```

[52]:
```
                    trigram
0          march all stories
1            all stories new
2            stories new and
3           new and complete
4     and complete publisher
5  complete publisher editor
6       publisher editor if
7              editor if is
8            if is published
9            is published bi
```

[53]:
```
# count, sort, pick to 10

from pyspark.sql.functions import desc
df.groupBy("trigram").count()\
    .orderBy(desc("count"))\
    .show()
```

```
[Stage 92:=============================================>           (4 + 1) / 5]

+------------+-----+
|     trigram|count|
+------------+-----+
|     i don t|16282|
|  one of the|10616|
|  out of the| 8441|
| there was a| 8050|
|  don t know| 7361|
|    it was a| 7318|
|    i didn t| 6040|
|there was no| 5834|
|   you don t| 5747|
|     i can t| 4783|
|   he didn t| 4560|
|     i m not| 4265|
```

```
|      it s a| 4215|
| back to the| 3961|
| it would be| 3915|
|    a lot of| 3882|
| the rest of| 3783|
|for a moment| 3754|
|  it was the| 3695|
| he had been| 3678|
+------------+-----+
only showing top 20 rows
```

[ ]:

[ ]: