Lab2_python

December 10, 2022

[3]:
```
import findspark
findspark.init()
```

[4]:
```
from pyspark import SparkConf
from pyspark import SparkContext
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[4]"))
```

[5]:
```
import random
flips = 1000000
coins = range(flips)
rdd_coins = sc.parallelize(coins)
rdd_flips = rdd_coins.map(lambda i: random.random())
rdd_heads = rdd_flips.filter(lambda r: r < 0.51 )
rdd_heads.count()
```

[5]: 509196

[6]:
```
import random
num_samples = 1000000
def inside(p):
    x,y = random.random(), random.random()
    return x*x + y*y < 1
count = sc.parallelize(range(0, num_samples)).filter(inside).count()
pi = 4* count/num_samples
print(pi)
```

```
3.141636
```

[7]:
```
from pyspark.sql.dataframe import DataFrame
from pyspark.sql import SparkSession
spark = SparkSession(sc)
```

[8]:
```
df = spark.read.format('com.databricks.spark.csv').\
                                options(header='true', \
                                inferschema='true').load("/home/spark/files/
 ↪gcredit.csv",header=True);
```

[9]:
```
df.columns
```

[9]:
```
['Creditability',
 'Account Balance',
 'Duration of Credit (month)',
 'Payment Status of Previous Credit',
 'Purpose',
 'Credit Amount',
 'Value Savings/Stocks',
 'Length of current employment',
 'Instalment per cent',
 'Sex & Marital Status',
 'Guarantors',
 'Duration in Current address',
 'Most valuable available asset',
 'Age (years)',
 'Concurrent Credits',
 'Type of apartment',
 'No of Credits at this Bank',
 'Occupation',
 'No of dependents',
 'Telephone',
 'Foreign Worker']
```

[10]:
```
df[['Account Balance','No of dependents']].toPandas().describe()
```

[10]:

	Account Balance	No of dependents
count	1000.000000	1000.000000
mean	2.577000	1.155000
std	1.257638	0.362086
min	1.000000	1.000000
25%	1.000000	1.000000
50%	2.000000	1.000000
75%	4.000000	1.000000
max	4.000000	2.000000

[11]:
```
def describe_pd(df_in, columns, style):
    '''
    Function to union the basic stats results and deciles
    :param df_in: the input dataframe
    :param columns: the cloumn name list of the numerical variable
    :param style: the display style

    :return : the numerical describe info. of the input dataframe

    :author: Wenqiang Feng
    :email:  von198@gmail.com
    '''
```

```
    if style == 1:
        percentiles = [25, 50, 75]
    else:
        percentiles = np.array(range(0, 110, 10))

    percs = np.transpose([np.percentile(df_in.select(x).collect(), percentiles)␣
↪for x in columns])
    percs = pd.DataFrame(percs, columns=columns)
    percs['summary'] = [str(p) + '%' for p in percentiles]

    spark_describe = df_in.describe().toPandas()
    new_df = pd.concat([spark_describe, percs],ignore_index=True,sort=True)
    new_df = new_df.round(2)
    return new_df[['summary'] + columns]
```

[12]:
```
num_cols = ['Account Balance','No of dependents']
```

[13]:
```
df.select(num_cols).describe().show()
```

```
+-------+-----------------+-------------------+
|summary|   Account Balance|   No of dependents|
+-------+-----------------+-------------------+
|  count|             1000|               1000|
|   mean|            2.577|              1.155|
| stddev|1.2576377271108936|0.36208577175319395|
|    min|                1|                  1|
|    max|                4|                  2|
+-------+-----------------+-------------------+
```

[14]:
```
import numpy as np
import pandas as pd
```

[15]:
```
output = describe_pd(df,num_cols,1)
```

[16]:
```
output['summary']= output['summary'].astype(str)
# convert just columns
output[num_cols] = output[num_cols].apply(pd.to_numeric)
```

[17]:
```
output.dtypes
```

[17]:
```
summary              object
Account Balance     float64
No of dependents    float64
dtype: object
```

[18]:
```
spark.createDataFrame(output).show()
```

```
+-------+------------------+-----------------+
|summary|   Account Balance| No of dependents|
+-------+------------------+-----------------+
|  count|            1000.0|           1000.0|
|   mean|             2.577|            1.155|
| stddev|1.2576377271108936|0.362085771753194|
|    min|               1.0|              1.0|
|    max|               4.0|              2.0|
|    25%|               1.0|              1.0|
|    50%|               2.0|              1.0|
|    75%|               4.0|              1.0|
+-------+------------------+-----------------+
```

[19]:
```
spark.createDataFrame(output).show()
```

```
+-------+------------------+-----------------+
|summary|   Account Balance| No of dependents|
+-------+------------------+-----------------+
|  count|            1000.0|           1000.0|
|   mean|             2.577|            1.155|
| stddev|1.2576377271108936|0.362085771753194|
|    min|               1.0|              1.0|
|    max|               4.0|              2.0|
|    25%|               1.0|              1.0|
|    50%|               2.0|              1.0|
|    75%|               4.0|              1.0|
+-------+------------------+-----------------+
```

[20]:
```
output['summary']= output['summary'].astype(str)
# convert just columns
output[num_cols] = output[num_cols].apply(pd.to_numeric)
spark.createDataFrame(output).show()
```

```
+-------+------------------+-----------------+
|summary|   Account Balance| No of dependents|
+-------+------------------+-----------------+
|  count|            1000.0|           1000.0|
|   mean|             2.577|            1.155|
| stddev|1.2576377271108936|0.362085771753194|
|    min|               1.0|              1.0|
|    max|               4.0|              2.0|
|    25%|               1.0|              1.0|
|    50%|               2.0|              1.0|
|    75%|               4.0|              1.0|
+-------+------------------+-----------------+
```

[21]:
```
var = 'Age (years)'
#  pyspark.sql.function
#df.select(skewness(var),kurtosis(var)).show()
#  pandas skew(), kurtosis()
df[['Age (years)']].toPandas().skew(),df[['Age (years)']].toPandas().kurtosis()
```

[21]:
```
(Age (years)    1.024712
 dtype: float64,
 Age (years)    0.620529
 dtype: float64)
```

[22]:
```
df.select('Credit Amount').show(5)
```

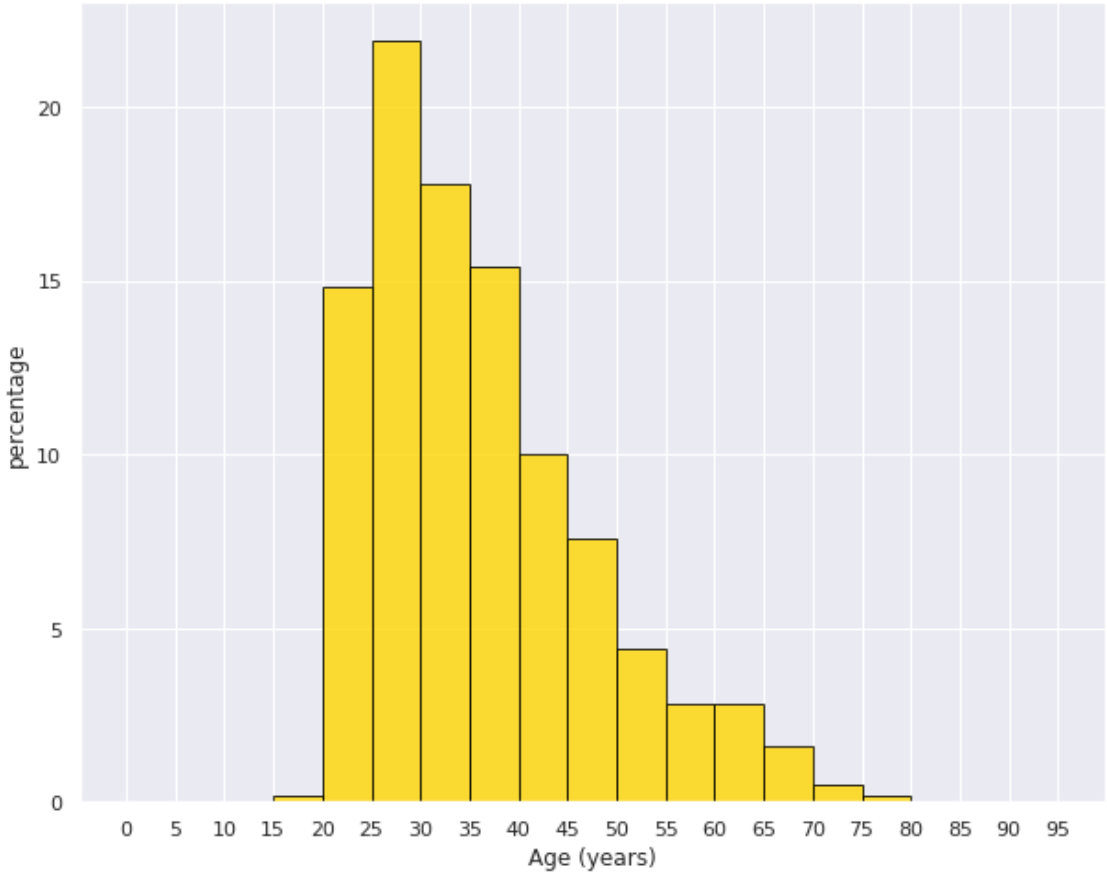
```
+-------------+
|Credit Amount|
+-------------+
|         1049|
|         2799|
|          841|
|         2122|
|         2171|
+-------------+
only showing top 5 rows
```

[23]:
```
data1 = df.select('Age (years)').toPandas()
```

[24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
%matplotlib inline

plt.rcParams['figure.figsize'] =(16,9)
plt.style.use('ggplot')
sns.set()
```

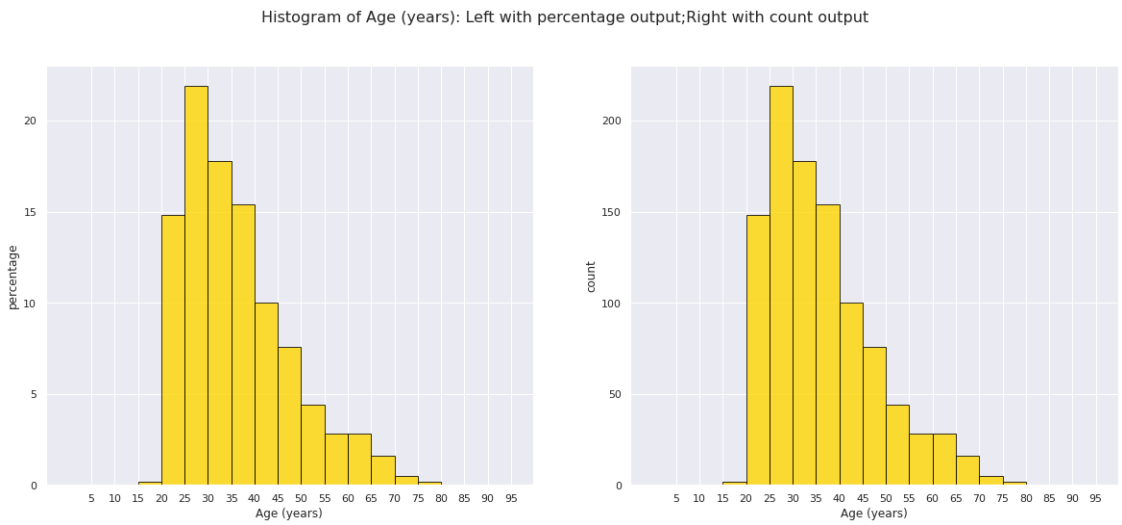
[25]:
```
var = 'Age (years)'
x = data1[var]
bins = np.arange(0, 100, 5.0)

plt.figure(figsize=(10,8))
# the histogram of the data
plt.hist(x, bins, alpha=0.8, histtype='bar', color='gold',
         ec='black',weights=np.zeros_like(x) + 100. / x.size)
```

```
plt.xlabel(var)
plt.ylabel('percentage')
plt.xticks(bins)
plt.show()

#fig.savefig(var+".pdf", bbox_inches='tight')
```



[26]:
```
var = 'Age (years)'
x = data1[var]
bins = np.arange(0, 100, 5.0)


###############################################################################
hist, bin_edges = np.histogram(x,bins,
                               weights=np.zeros_like(x) + 100. / x.size)
# make the histogram

fig = plt.figure(figsize=(20, 8))
ax = fig.add_subplot(1, 2, 1)
```

```
# Plot the histogram heights against integers on the x axis
ax.bar(range(len(hist)),hist,width=1,alpha=0.8,ec ='black', color='gold')
# # Set the ticks to the middle of the bars
ax.set_xticks([0.5+i for i,j in enumerate(hist)])
# Set the xticklabels to a string that tells us what the bin edges were
labels =['{}'.format(int(bins[i+1])) for i,j in enumerate(hist)]
#labels.insert(0,'0')
ax.set_xticklabels(labels)
plt.xlabel(var)
plt.ylabel('percentage')

##########################################################################

hist, bin_edges = np.histogram(x,bins) # make the histogram

ax = fig.add_subplot(1, 2, 2)
# Plot the histogram heights against integers on the x axis
ax.bar(range(len(hist)),hist,width=1,alpha=0.8,ec ='black', color='gold')

# # Set the ticks to the middle of the bars
ax.set_xticks([0.5+i for i,j in enumerate(hist)])

# Set the xticklabels to a string that tells us what the bin edges were
labels =['{}'.format(int(bins[i+1])) for i,j in enumerate(hist)]
#labels.insert(0,'0')
ax.set_xticklabels(labels)
plt.xlabel(var)
plt.ylabel('count')
plt.suptitle('Histogram of {}: Left with percentage output;Right with count␣
 ↪output'
             .format(var), size=16)
plt.show()

#fig.savefig(var+".pdf", bbox_inches='tight')
```



[27]:
```
def age_condition(x):
    if pd.isnull(x):
        return "missing"
    elif x < 25:
        return "<25"
    elif 25 <= x <= 34:
        return "25-34"
    elif 35 <= x <= 44:
        return "35-44"
    elif 45 <= x <= 54:
        return "45-54"
    elif 55 <= x <= 64:
        return "55-64"
    else:
        return "65+"
```

[28]:
```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType, DoubleType

age_udf = udf(lambda x: age_condition(x), StringType())
```

[29]:
```
df = df.withColumn("age_class", age_udf("Age (years)"))
```

[30]:
```
df.select(['age_class','Age (years)']).show(3)
```

```
+---------+-----------+
|age_class|Age (years)|
+---------+-----------+
|      <25|         21|
|    35-44|         36|
```

```
|      <25|         23|
+---------+-----------+
only showing top 3 rows
```

[31]:
```
df.select(['age_class','Credit Amount']).\
   groupBy('age_class').count().show()
```

```
+---------+-----+
|age_class|count|
+---------+-----+
|    45-54|  120|
|      <25|  150|
|    55-64|   56|
|    35-44|  254|
|    25-34|  397|
|      65+|   23|
+---------+-----+
```

[32]:
```
df.stat.crosstab("age_class", "Occupation").show()
```

```
+--------------------+---+---+---+---+
|age_class_Occupation|  1|  2|  3|  4|
+--------------------+---+---+---+---+
|                 <25|  4| 34|108|  4|
|               55-64|  1| 15| 31|  9|
|               25-34|  7| 61|269| 60|
|               35-44|  4| 58|143| 49|
|                 65+|  5|  3|  6|  9|
|               45-54|  1| 29| 73| 17|
+--------------------+---+---+---+---+
```

[33]:
```
from pyspark.sql import functions as F
from pyspark.sql.functions import rank,sum,col
from pyspark.sql import Window

window = Window.rowsBetween(Window.unboundedPreceding,Window.unboundedFollowing)
# withColumn('Percent %',F.format_string("%5.0f%%\n",col('Credit_num')*100/
 ↪col('total'))).\
tab = df.select(['age_class','Credit Amount']).\
   groupBy('age_class').\
   agg(F.count('Credit Amount').alias('Credit_num'),
       F.mean('Credit Amount').alias('Credit_avg'),
       F.min('Credit Amount').alias('Credit_min'),
       F.max('Credit Amount').alias('Credit_max')).\
   withColumn('total',sum(col('Credit_num')).over(window)).\
```

```
   withColumn('Percent',col('Credit_num')*100/col('total')).\
   drop(col('total'))
```

[34]:
```
tab.show()
```

```
+---------+----------+------------------+----------+----------+-------+
|age_class|Credit_num|        Credit_avg|Credit_min|Credit_max|Percent|
+---------+----------+------------------+----------+----------+-------+
|    45-54|       120|3183.0666666666666|       338|     12612|   12.0|
|      <25|       150| 2970.733333333333|       276|     15672|   15.0|
|    55-64|        56| 3493.660714285714|       385|     15945|    5.6|
|    35-44|       254| 3403.771653543307|       250|     15857|   25.4|
|    25-34|       397| 3298.823677581864|       343|     18424|   39.7|
|      65+|        23|3210.1739130434785|       571|     14896|    2.3|
+---------+----------+------------------+----------+----------+-------+
```

[35]:
```
plot_data = tab.toPandas()
plot_data.sort_values('age_class')
```

[35]:

	age_class	Credit_num	Credit_avg	Credit_min	Credit_max	Percent
4	25-34	397	3298.823678	343	18424	39.7
3	35-44	254	3403.771654	250	15857	25.4
0	45-54	120	3183.066667	338	12612	12.0
2	55-64	56	3493.660714	385	15945	5.6
5	65+	23	3210.173913	571	14896	2.3
1	<25	150	2970.733333	276	15672	15.0

[36]:
```
custom_dict = {'<25': 0, '25-34': 1, '35-44': 2, '45-54': 3, '55-64': 4, '65+':␣
↪5}
```

[37]:
```
plot_data['index']= plot_data['age_class'].replace(custom_dict)
```

[38]:
```
plot_data.sort_values('index')
```

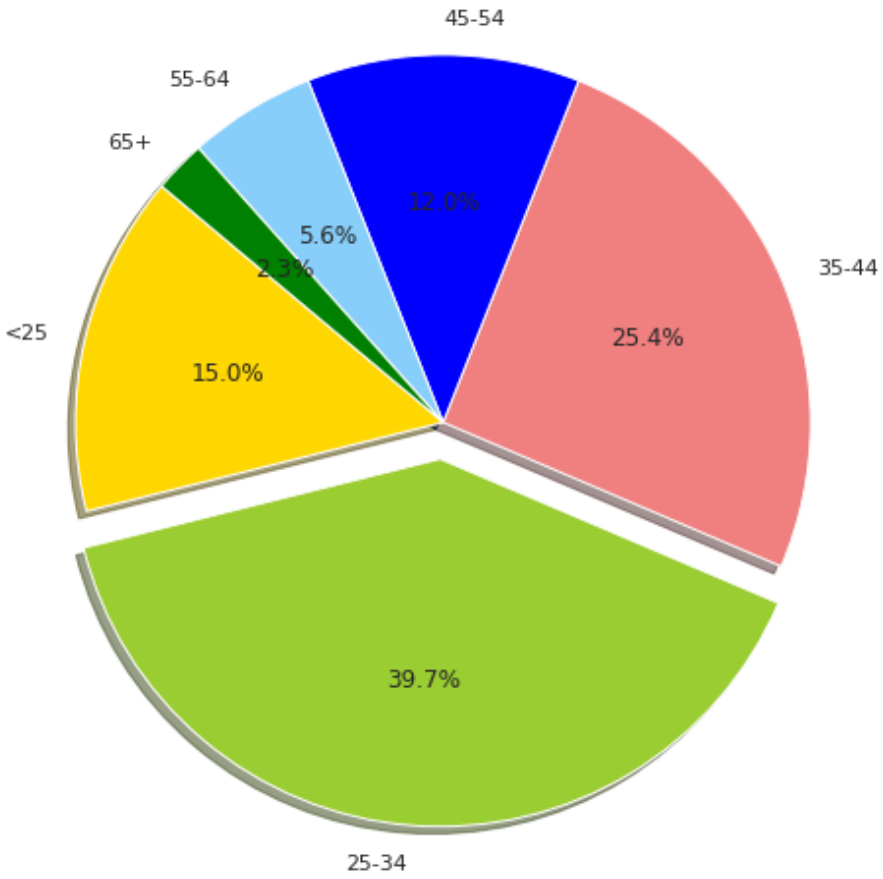
[38]:

	age_class	Credit_num	Credit_avg	Credit_min	Credit_max	Percent	index
1	<25	150	2970.733333	276	15672	15.0	0
4	25-34	397	3298.823678	343	18424	39.7	1
3	35-44	254	3403.771654	250	15857	25.4	2
0	45-54	120	3183.066667	338	12612	12.0	3
2	55-64	56	3493.660714	385	15945	5.6	4
5	65+	23	3210.173913	571	14896	2.3	5

[39]:
```
plot_data = plot_data.sort_values('index')
```

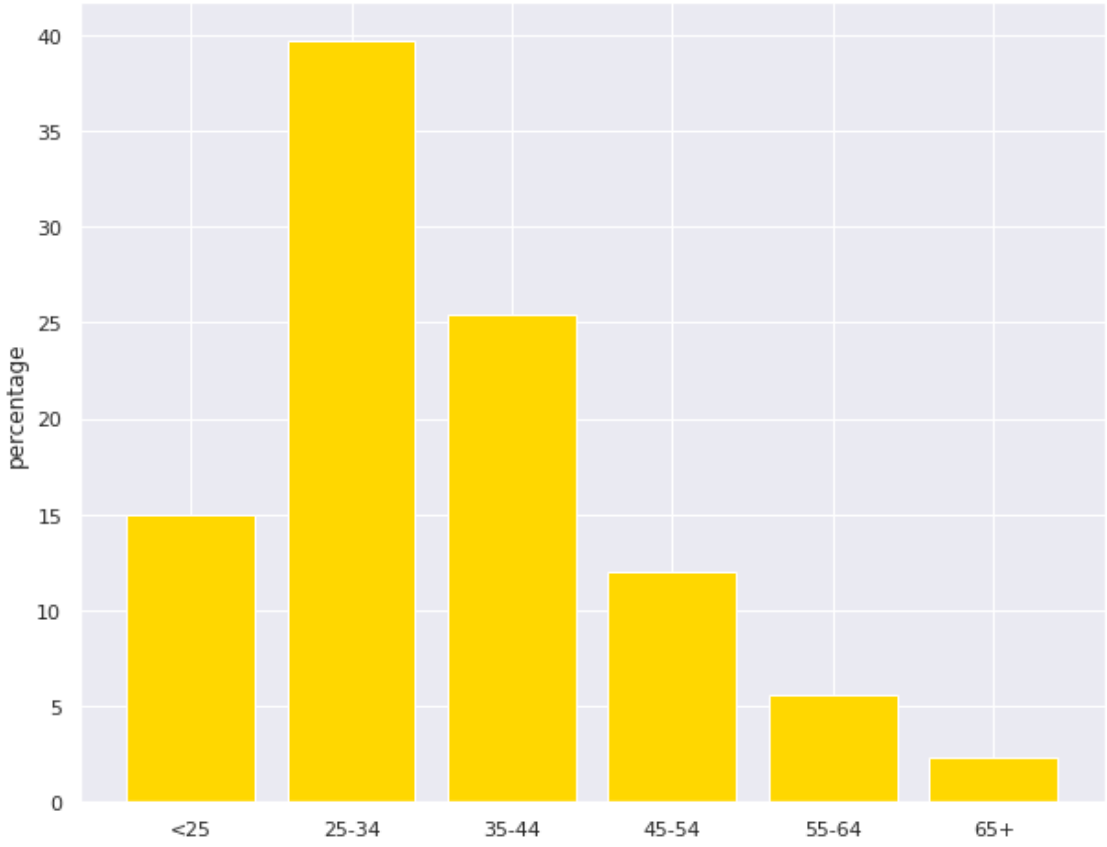
[40]:
```
# Data to plot
labels = plot_data.age_class
```

```
sizes =  plot_data.Percent
colors = ['gold', 'yellowgreen', 'lightcoral','blue',␣
 ↪'lightskyblue','green','red']
explode = (0, 0.1, 0, 0,0,0)  # explode 1st slice

# Plot
plt.figure(figsize=(10,8))
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```



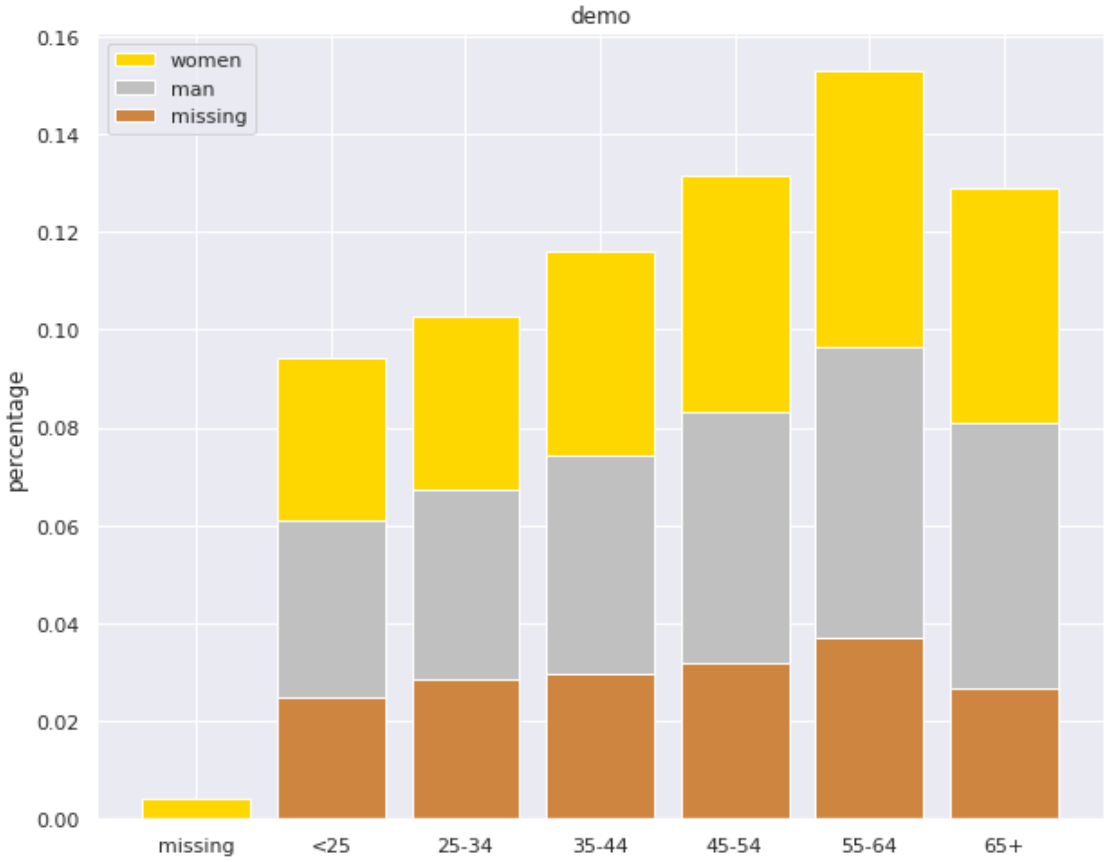
[41]:
```
labels = plot_data.age_class
missing = plot_data.Percent
ind = [x for x, _ in enumerate(labels)]

plt.figure(figsize=(10,8))
plt.bar(ind, missing, width=0.8, label='missing', color='gold')
```

```
plt.xticks(ind, labels)
plt.ylabel("percentage")

plt.show()
```



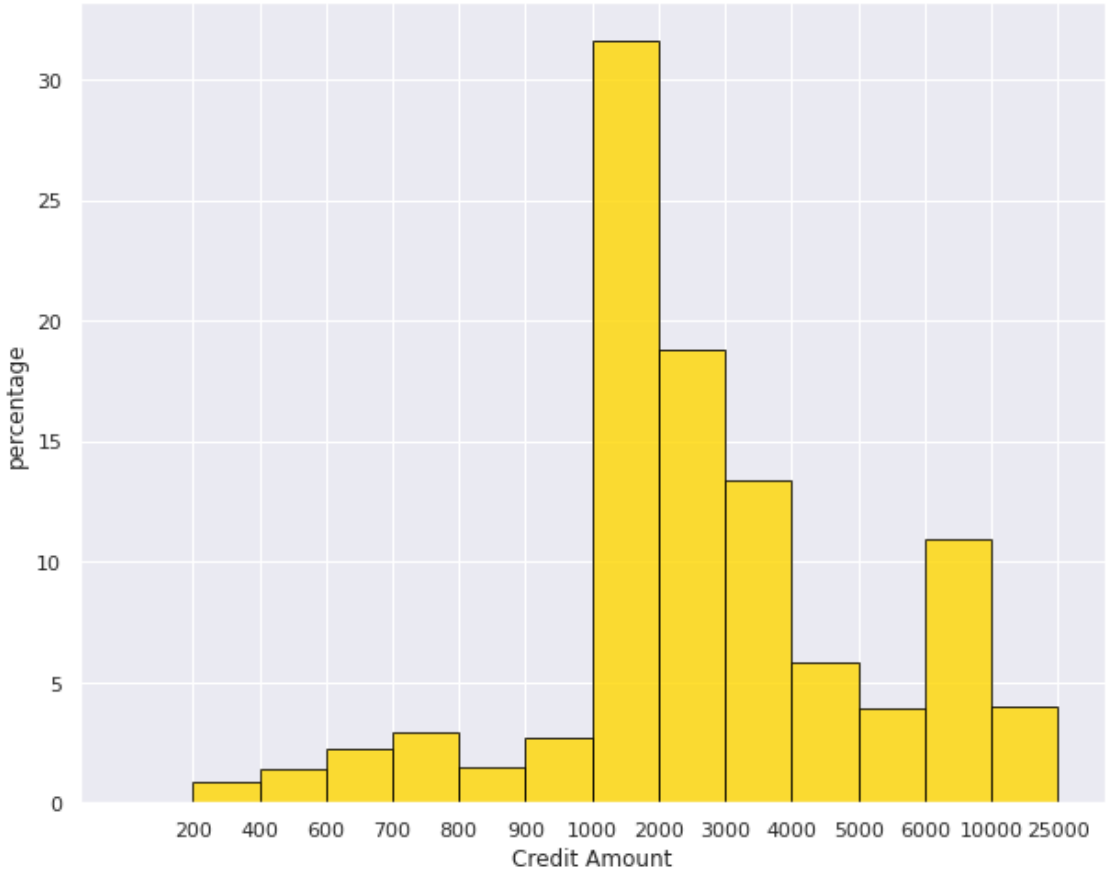
[42]:
```
labels = ['missing', '<25', '25-34', '35-44', '45-54','55-64','65+']
missing = np.array([0.000095, 0.024830, 0.028665, 0.029477, 0.031918,0.037073,0.
↪026699])
man = np.array([0.000147, 0.036311, 0.038684, 0.044761, 0.051269, 0.059542, 0.
↪054259])
women = np.array([0.004035, 0.032935, 0.035351, 0.041778, 0.048437, 0.056236,0.
↪048091])
ind = [x for x, _ in enumerate(labels)]

plt.figure(figsize=(10,8))
plt.bar(ind, women, width=0.8, label='women', color='gold', bottom=man+missing)
plt.bar(ind, man, width=0.8, label='man', color='silver', bottom=missing)
plt.bar(ind, missing, width=0.8, label='missing', color='#CD853F')
```

```
plt.xticks(ind, labels)
plt.ylabel("percentage")
plt.legend(loc="upper left")
plt.title("demo")

plt.show()
```



[43]:
```
# prepare for the plot data

var = 'Credit Amount'
plot_data = df.select(var).toPandas()
x= plot_data[var]

bins =[0,200,400,600,700,800,900,1000,2000,3000,4000,5000,6000,10000,25000]

hist, bin_edges = np.histogram(x,bins,weights=np.zeros_like(x) + 100. / x.size)␣
 ↪# make the histogram

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(1, 1, 1)
```

```
# Plot the histogram heights against integers on the x axis
ax.bar(range(len(hist)),hist,width=1,alpha=0.8,ec ='black',color = 'gold')

# # Set the ticks to the middle of the bars
ax.set_xticks([0.5+i for i,j in enumerate(hist)])

# Set the xticklabels to a string that tells us what the bin edges were
#labels =['{}k'.format(int(bins[i+1]/1000)) for i,j in enumerate(hist)]
labels =['{}'.format(bins[i+1]) for i,j in enumerate(hist)]
#labels.insert(0,'0')
ax.set_xticklabels(labels)
#plt.text(-0.6, -1.4,'0')
plt.xlabel(var)
plt.ylabel('percentage')
plt.show()
```



[44]:
```
################################################
```

[45]:
```
import findspark
findspark.init()

# configure spark variables
from pyspark.context import SparkContext
from pyspark import SparkConf
from pyspark.sql.context import SQLContext
from pyspark.sql.session import SparkSession

sqlContext = SQLContext(sc)
spark = SparkSession(sc)

# load up other dependencies
import re
import pandas as pd
```

[46]:
```
m = re.finditer(r'.*?(spark).*?', "I'm searching for a spark in PySpark", re.I)
for match in m:
    print(match, match.start(), match.end())
```

```
<re.Match object; span=(0, 25), match="I'm searching for a spark"> 0 25
<re.Match object; span=(25, 36), match=' in PySpark'> 25 36
```

[47]:
```
raw_data_files = ['/home/spark/files/access_log1', '/home/spark/files/
 ↪access_log2']
base_df = spark.read.text(raw_data_files)
base_df.printSchema()
```

```
root
 |-- value: string (nullable = true)
```

[48]:
```
type(base_df)
```

[48]:
```
pyspark.sql.dataframe.DataFrame
```

[49]:
```
base_df_rdd = base_df.rdd
type(base_df_rdd)
```

[49]:
```
pyspark.rdd.RDD
```

[50]:
```
base_df.show(10, truncate=False)
```

```
+------------------------------------------------------------------------------
------------------------------------------------------------------------------
-----------------------------------------------------------------------+
|value
|
```

```
+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------+
|2001:6d8:10:4400:451:aebb:715b:b6df - - [10/Nov/2019:03:44:13 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"
|
|2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:13 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"|
|172.30.254.52 - - [10/Nov/2019:03:44:15 +0100] "GET /wpad.dat HTTP/1.1" 200 137
"-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"            |
|2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:21 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"|
|172.30.254.52 - - [10/Nov/2019:03:44:23 +0100] "GET /wpad.dat HTTP/1.1" 200 137
"-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"            |
|2001:6d8:10:4401:58:1a77:a67:aaef - - [10/Nov/2019:03:44:28 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "WinHttp-Autoproxy-Service/5.1"
|
|2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:29 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"|
|172.30.254.52 - - [10/Nov/2019:03:44:31 +0100] "GET /wpad.dat HTTP/1.1" 200 137
"-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"            |
|2001:6d8:10:4400:b59f:6067:d7c5:2b84 - - [10/Nov/2019:03:44:32 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"
|
|2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:37 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"|
+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------+
only showing top 10 rows
```

[51]:
```
base_df_rdd.take(10)
```

[51]:
```
[Row(value='2001:6d8:10:4400:451:aebb:715b:b6df - - [10/Nov/2019:03:44:13 +0100]
"GET /wpad.dat HTTP/1.1" 304 - "-" "-"'),
 Row(value='2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:13 +0100] "GET
/wpad.dat HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) ReaderServices/19.12.20036
Chrome/80.0.0.0 Safari/537.36"'),
```

```
 Row(value='172.30.254.52 - - [10/Nov/2019:03:44:15 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"'),
 Row(value='2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:21 +0100] "GET
/wpad.dat HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) ReaderServices/19.12.20036
Chrome/80.0.0.0 Safari/537.36"'),
 Row(value='172.30.254.52 - - [10/Nov/2019:03:44:23 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"'),
 Row(value='2001:6d8:10:4401:58:1a77:a67:aaef - - [10/Nov/2019:03:44:28 +0100]
"GET /wpad.dat HTTP/1.1" 304 - "-" "WinHttp-Autoproxy-Service/5.1"'),
 Row(value='2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:29 +0100] "GET
/wpad.dat HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) ReaderServices/19.12.20036
Chrome/80.0.0.0 Safari/537.36"'),
 Row(value='172.30.254.52 - - [10/Nov/2019:03:44:31 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"'),
 Row(value='2001:6d8:10:4400:b59f:6067:d7c5:2b84 - - [10/Nov/2019:03:44:32
+0100] "GET /wpad.dat HTTP/1.1" 304 - "-" "-"'),
 Row(value='2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:37 +0100] "GET
/wpad.dat HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) ReaderServices/19.12.20036
Chrome/80.0.0.0 Safari/537.36"')]
```

[52]:
```
print((base_df.count(), len(base_df.columns)))
```

```
(285061, 1)
```

[53]:
```
sample_logs = [item['value'] for item in base_df.take(15)]
sample_logs
```

[53]:
```
['2001:6d8:10:4400:451:aebb:715b:b6df - - [10/Nov/2019:03:44:13 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"',
 '2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:13 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '172.30.254.52 - - [10/Nov/2019:03:44:15 +0100] "GET /wpad.dat HTTP/1.1" 200
137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:21 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '172.30.254.52 - - [10/Nov/2019:03:44:23 +0100] "GET /wpad.dat HTTP/1.1" 200
137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
```

```
 '2001:6d8:10:4401:58:1a77:a67:aaef - - [10/Nov/2019:03:44:28 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "WinHttp-Autoproxy-Service/5.1"',
 '2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:29 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '172.30.254.52 - - [10/Nov/2019:03:44:31 +0100] "GET /wpad.dat HTTP/1.1" 200
137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84 - - [10/Nov/2019:03:44:32 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"',
 '2001:6d8:10:4400::1:100b - - [10/Nov/2019:03:44:37 +0100] "GET /wpad.dat
HTTP/1.1" 200 137 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84 - - [10/Nov/2019:03:44:39 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"',
 '172.30.254.52 - - [10/Nov/2019:03:44:39 +0100] "GET /wpad.dat HTTP/1.1" 200
137 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) ReaderServices/19.12.20036 Chrome/80.0.0.0 Safari/537.36"',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84 - - [10/Nov/2019:03:44:40 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"',
 '2001:6d8:10:4400:5146:1e50:99:5f7b - - [10/Nov/2019:03:44:41 +0100] "GET
/wpad.dat HTTP/1.1" 200 137 "-" "-"',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84 - - [10/Nov/2019:03:44:41 +0100] "GET
/wpad.dat HTTP/1.1" 304 - "-" "-"']
```

[54]:
```
host_pattern = r'(^\S+)\s'
hosts = [re.search(host_pattern, item).group(1)
           if re.search(host_pattern, item)
           else 'no match'
           for item in sample_logs]
hosts
```

[54]:
```
['2001:6d8:10:4400:451:aebb:715b:b6df',
 '2001:6d8:10:4400::1:100b',
 '172.30.254.52',
 '2001:6d8:10:4400::1:100b',
 '172.30.254.52',
 '2001:6d8:10:4401:58:1a77:a67:aaef',
 '2001:6d8:10:4400::1:100b',
 '172.30.254.52',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84',
 '2001:6d8:10:4400::1:100b',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84',
 '172.30.254.52',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84',
 '2001:6d8:10:4400:5146:1e50:99:5f7b',
 '2001:6d8:10:4400:b59f:6067:d7c5:2b84']
```

[55]:
```
ts_pattern = r'\[(\d{2}/\w{3}/\d{4}:\d{2}:\d{2}:\d{2} \S{1}\d{4})]'
timestamps = [re.search(ts_pattern, item).group(1)
                  if re.search(ts_pattern, item)
                  else 'no match'
                  for item in sample_logs]
timestamps
```

[55]:
```
['10/Nov/2019:03:44:13 +0100',
 '10/Nov/2019:03:44:13 +0100',
 '10/Nov/2019:03:44:15 +0100',
 '10/Nov/2019:03:44:21 +0100',
 '10/Nov/2019:03:44:23 +0100',
 '10/Nov/2019:03:44:28 +0100',
 '10/Nov/2019:03:44:29 +0100',
 '10/Nov/2019:03:44:31 +0100',
 '10/Nov/2019:03:44:32 +0100',
 '10/Nov/2019:03:44:37 +0100',
 '10/Nov/2019:03:44:39 +0100',
 '10/Nov/2019:03:44:39 +0100',
 '10/Nov/2019:03:44:40 +0100',
 '10/Nov/2019:03:44:41 +0100',
 '10/Nov/2019:03:44:41 +0100']
```

[56]:
```
method_uri_protocol_pattern = r'\"(\S+)\s(\S+)\s*(\S*)\"'
method_uri_protocol = [re.search(method_uri_protocol_pattern, item).groups()
                if re.search(method_uri_protocol_pattern, item)
                else 'no match'
               for item in sample_logs]
method_uri_protocol
```

[56]:
```
[('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1'),
 ('GET', '/wpad.dat', 'HTTP/1.1')]
```

[57]:
```
from pyspark.sql.functions import regexp_extract

logs_df = base_df.select(regexp_extract('value', r'(^[\S+]+) -', 1).
 ↪alias('host'),
                         regexp_extract('value', r'(^[\S+]+) - - \[(.*)\]', 2).
 ↪alias('timestamp'),
                         regexp_extract('value', r'(^[\S+]+) - - \[(.*)\]␣
 ↪"(\w+)', 3).alias('method'),
                         regexp_extract('value', r'(^[\S+]+) - - \[(.*)\]␣
 ↪"(\w+) (.*?) (.*?)', 4).alias('path'),
                         regexp_extract('value', r'(^[\S+]+) - - \[(.*) \+(.
 ↪*)\] "(\w+) (.*?) (.*?)" ', 6).alias('protocol'),
                         regexp_extract('value', r'(^[\S+]+) - - \[(.*) \+(.
 ↪*)\] "(\w+) (.*?) (.*?)" (\d+) ', 7).cast('integer').alias('code'),
                         regexp_extract('value', r'(^[\S+]+) - - \[(.*) \+(.
 ↪*)\] "(\w+) (.*?) (.*?)" (\d+) (\d+)', 8).cast('integer').alias('size'))
logs_df.show(10, truncate=True)
print((logs_df.count(), len(logs_df.columns)))
```

```
+--------------------+--------------------+------+---------+--------+----+----+
|                host|           timestamp|method|     path|protocol|code|size|
+--------------------+--------------------+------+---------+--------+----+----+
|2001:6d8:10:4400:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 304|null|
|2001:6d8:10:4400:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
|       172.30.254.52|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
|2001:6d8:10:4400:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
|       172.30.254.52|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
|2001:6d8:10:4401:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 304|null|
|2001:6d8:10:4400:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
|       172.30.254.52|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
|2001:6d8:10:4400:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 304|null|
|2001:6d8:10:4400:…|10/Nov/2019:03:44…|   GET|/wpad.dat|HTTP/1.1| 200| 137|
+--------------------+--------------------+------+---------+--------+----+----+
only showing top 10 rows

(285061, 7)
```

[58]:
```
(base_df
    .filter(base_df['value']
                .isNull())
    .count())
```

[58]:
```
0
```

[59]:
```
bad_rows_df = logs_df.filter(logs_df['host'].isNull()|
#                             logs_df['timestamp'].isNull() |
#                             logs_df['method'].isNull() |
```

```
#                                logs_df['path'].isNull() |
                               logs_df['code'].isNull() |
                               logs_df['size'].isNull()|
                               logs_df['protocol'].isNull())
bad_rows_df.count()
```

[59]: 175714

[60]:
```
logs_df = logs_df.na.fill({'size': 0})
bad_rows_df.count()
```

[60]: 175714

[61]:
```
from pyspark.sql.functions import udf

month_map = {
  'Jan': 1, 'Feb': 2, 'Mar':3, 'Apr':4, 'May':5, 'Jun':6, 'Jul':7,
  'Aug':8,  'Sep': 9, 'Oct':10, 'Nov': 11, 'Dec': 12
}

def parse_clf_time(text):
    """ Convert Common Log time format into a Python datetime object
    Args:
        text (str): date and time in Apache time format [dd/mmm/yyyy:hh:mm:ss␣
 ↪(+/-)zzzz]
    Returns:
        a string suitable for passing to CAST('timestamp')
    """
    # NOTE: We're ignoring the time zones here, might need to be handled␣
 ↪depending on the problem you are solving
    return "{0:04d}-{1:02d}-{2:02d} {3:02d}:{4:02d}:{5:02d}".format(
      int(text[7:11]),
      month_map[text[3:6]],
      int(text[0:2]),
      int(text[12:14]),
      int(text[15:17]),
      int(text[18:20])
    )
```

[62]:
```
udf_parse_time = udf(parse_clf_time)

logs_df = (logs_df.select('*', udf_parse_time(logs_df['timestamp'])
                                    .cast('timestamp')
                                    .alias('time'))
                                    .drop('timestamp'))
```

[63]:
```
logs_df.printSchema()
```

```
root
 |-- host: string (nullable = true)
 |-- method: string (nullable = true)
 |-- path: string (nullable = true)
 |-- protocol: string (nullable = true)
 |-- code: integer (nullable = true)
 |-- size: integer (nullable = false)
 |-- time: timestamp (nullable = true)
```

[64]:
```
import numpy as np
import pandas as pd
```

[65]:
```
content_size_summary_df = logs_df.describe(['size'])
```

[66]:
```
content_size_summary_df.toPandas()
```

[66]:

	summary	size
0	count	285061
1	mean	4638.948316325278
2	stddev	360584.8484784875
3	min	0
4	max	82117397

[67]:
```
from pyspark.sql import functions as F

(logs_df.agg(F.min(logs_df['size']).alias('min_content_size'),
             F.max(logs_df['size']).alias('max_content_size'),
             F.mean(logs_df['size']).alias('mean_content_size'),
             F.stddev(logs_df['size']).alias('std_content_size'),
             F.count(logs_df['size']).alias('count_content_size'))
        .toPandas())
```

[67]:

	min_content_size	max_content_size	mean_content_size	std_content_size	count_content_size
0	0	82117397	4638.948316	360584.848478	285061

[68]:
```
status_freq_df = (logs_df
                     .groupBy('code')
                     .count()
                     .sort('code')
                     .cache())
print('Total distinct HTTP Status Codes:', status_freq_df.count())
```

```
Total distinct HTTP Status Codes: 10
```

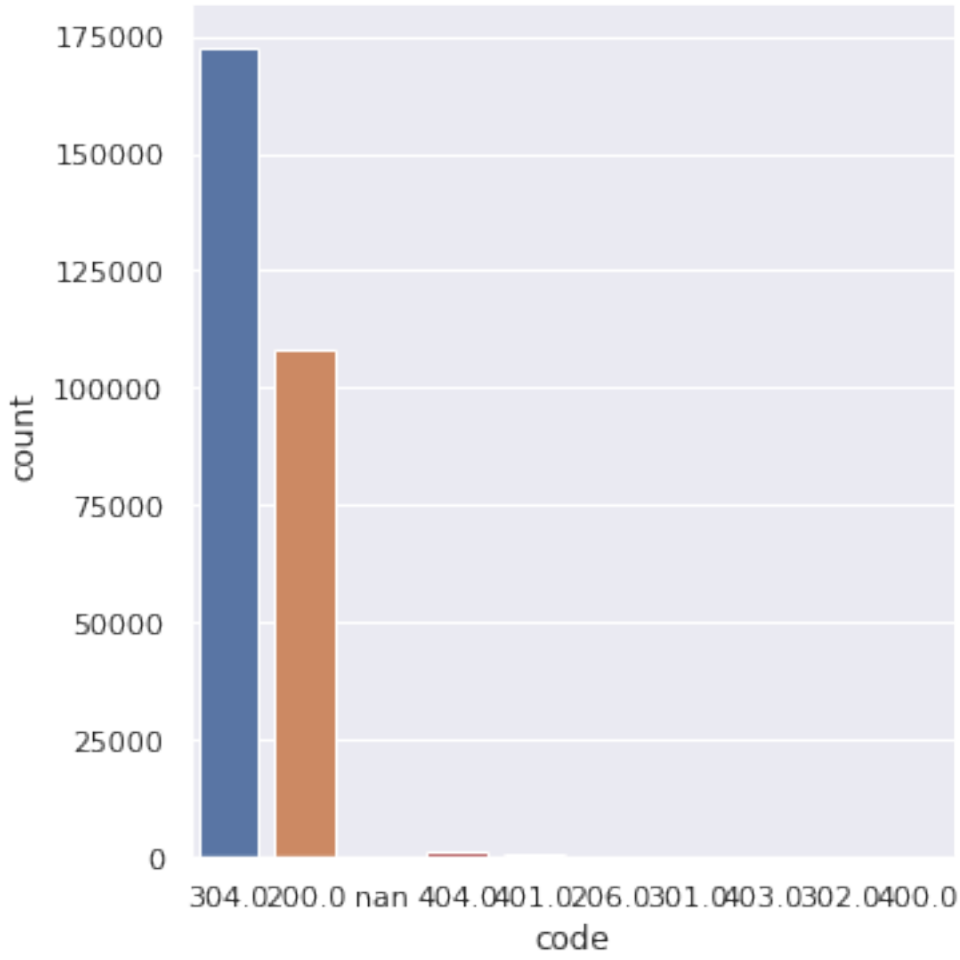
[69]:
```
status_freq_pd_df = (status_freq_df
                         .toPandas()
                         .sort_values(by=['count'],
                                      ascending=False))
status_freq_pd_df
```

[69]:

	code	count
5	304.0	172698
1	200.0	108149
0	NaN	2068
9	404.0	1450
7	401.0	508
2	206.0	65
3	301.0	52
8	403.0	37
4	302.0	33
6	400.0	1

[70]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

sns.catplot(x='code', y='count', data=status_freq_pd_df,
            kind='bar', order=status_freq_pd_df['code'])
```

[70]:
```
<seaborn.axisgrid.FacetGrid at 0x7f08416ddbb0>
```



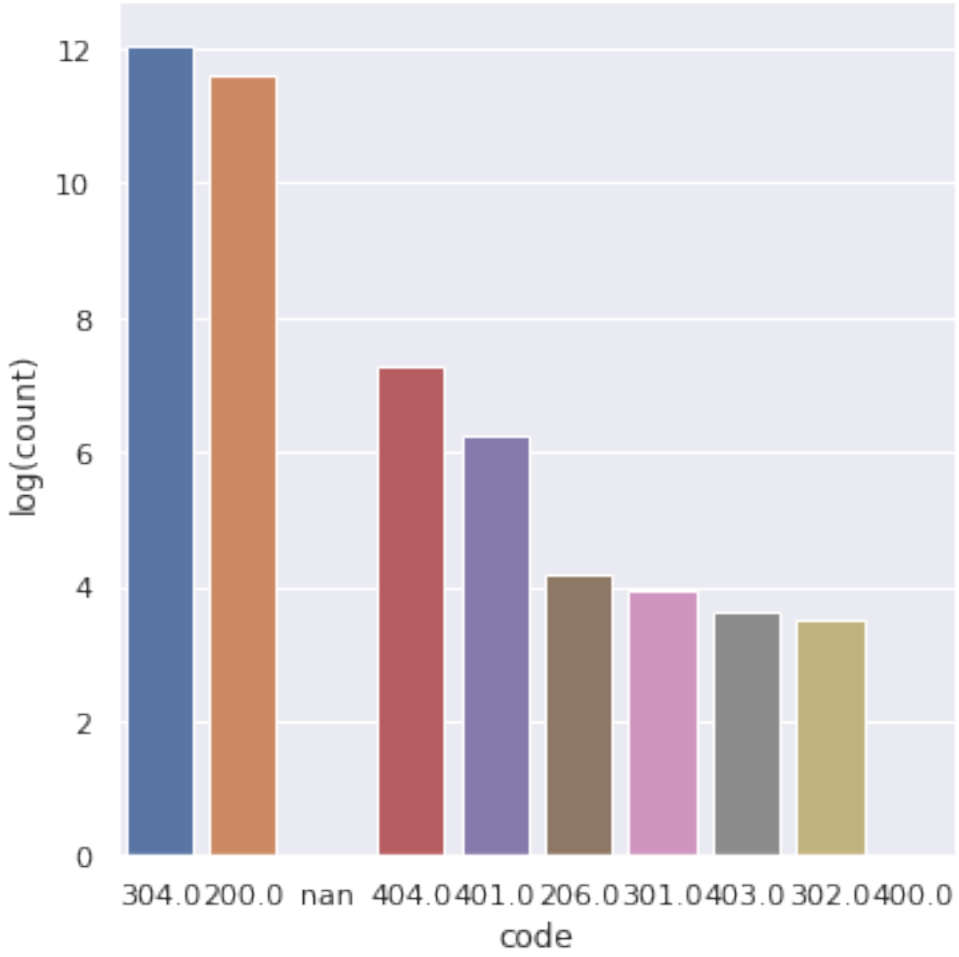
[71]:
```
log_freq_df = status_freq_df.withColumn('log(count)',
                                        F.log(status_freq_df['count']))
log_freq_df.show()
```

```
+----+------+------------------+
|code| count|        log(count)|
+----+------+------------------+
|null|  2068|  7.63433723562832|
| 200|108149|11.591265184921443|
| 206|    65| 4.174387269895637|
| 301|    52|3.9512437185814275|
| 302|    33|3.4965075614664802|
| 304|172698|12.059299683291801|
| 400|     1|               0.0|
| 401|   508| 6.230481447578482|
| 403|    37|3.6109179126442243|
| 404|  1450|  7.27931883541462|
```

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

[72]:
```
log_freq_pd_df = (log_freq_df
                    .toPandas()
                    .sort_values(by=['log(count)'],
                                 ascending=False))
sns.catplot(x='code', y='log(count)', data=log_freq_pd_df,
            kind='bar', order=status_freq_pd_df['code'])
```

[72]:
```
<seaborn.axisgrid.FacetGrid at 0x7f0841760d90>
```



[73]:
```
#ZADANIE DOMOWE␣
↪-------------------------------------------------------------------------------------------
```

[88]:
```
# Przyklad 1
```

[74]:
```
arrayStructureData = [
    [("James","","Smith"),("Java","Scala","C++"),"OH","M"],
    [("Anna","Rose",""),("Spark","Java","C++"),"NY","F"],
    [("Julia","","Williams"),("CSharp","VB"),"OH","F"],
    [("Maria","Anne","Jones"),("CSharp","VB"),"NY","M"],
    [("Jen","Mary","Brown"),("CSharp","VB"),"NY","M"],
    [("Mike","Mary","Williams"),("Python","VB"),"OH","M"]
]
arrayStructureData
```

[74]:
```
[[('James', '', 'Smith'), ('Java', 'Scala', 'C++'), 'OH', 'M'],
 [('Anna', 'Rose', ''), ('Spark', 'Java', 'C++'), 'NY', 'F'],
 [('Julia', '', 'Williams'), ('CSharp', 'VB'), 'OH', 'F'],
 [('Maria', 'Anne', 'Jones'), ('CSharp', 'VB'), 'NY', 'M'],
 [('Jen', 'Mary', 'Brown'), ('CSharp', 'VB'), 'NY', 'M'],
 [('Mike', 'Mary', 'Williams'), ('Python', 'VB'), 'OH', 'M']]
```

[76]:
```
from pyspark.sql.types import StructType, StringType, StructField, IntegerType,␣
 ↪ArrayType

nameStructType = StructType()
nameStructType.add("firstname",StringType())
nameStructType.add("middlename",StringType())
nameStructType.add("lastname",StringType())

arrayStructureSchema = StructType()
arrayStructureSchema.add("name",nameStructType)
arrayStructureSchema.add("languages", ArrayType(StringType()))
arrayStructureSchema.add("state", StringType())
arrayStructureSchema.add("gender", StringType())
```

[76]:
```
StructType(List(StructField(name,StructType(List(StructField(firstname,StringTyp
e,true),StructField(middlename,StringType,true),StructField(lastname,StringType,
true))),true),StructField(languages,ArrayType(StringType,true),true),StructField
(state,StringType,true),StructField(gender,StringType,true)))
```

[78]:
```
df = spark.createDataFrame(spark.sparkContext.
 ↪parallelize(arrayStructureData),arrayStructureSchema)
```

[79]:
```
df.filter(df.state == "OH").show()
df.filter("state == 'OH'").show()
df.filter(col("state") == "OH").show()
df.where(df.state == "OH").show()
df.where("state == 'OH'").show()
df.where(col("state") == "OH").show()
```

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

```
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
| [Julia, , Williams]|      [CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+

+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
| [Julia, , Williams]|      [CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+

+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
| [Julia, , Williams]|      [CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+

+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
| [Julia, , Williams]|      [CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+

+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
| [Julia, , Williams]|      [CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+

+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
| [Julia, , Williams]|      [CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

[81]:
```
df.filter("gender == 'M'").show()
df.where("gender == 'M'").show()
```

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
|[Maria, Anne, Jones]|      [CSharp, VB]|   NY|     M|
|  [Jen, Mary, Brown]|      [CSharp, VB]|   NY|     M|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+-----------------+-----+------+

+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
|[Maria, Anne, Jones]|      [CSharp, VB]|   NY|     M|
|  [Jen, Mary, Brown]|      [CSharp, VB]|   NY|     M|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+-----------------+-----+------+
```

[86]:
```
df.filter((df.state == "OH") & (df.gender == "M")).show()
df.where((df.state == "OH") & (df.gender == "M")).show()
```

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+-----------------+-----+------+

+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|    [James, , Smith]|[Java, Scala, C++]|   OH|     M|
|[Mike, Mary, Will…|      [Python, VB]|   OH|     M|
+--------------------+-----------------+-----+------+
```

[87]:
```
from pyspark.sql.functions import array_contains
df.filter(array_contains(df.languages,"Java")).show()
df.filter(df.name.lastname == "Williams").show()
```

```
+---------------+-----------------+-----+------+
|           name|        languages|state|gender|
+---------------+-----------------+-----+------+
|[James, , Smith]|[Java, Scala, C++]|   OH|     M|
```

```
|  [Anna, Rose, ]|[Spark, Java, C++]|   NY|     F|
+----------------+------------------+-----+------+

+--------------------+------------+-----+------+
|                name|   languages|state|gender|
+--------------------+------------+-----+------+
| [Julia, , Williams]|[CSharp, VB]|   OH|     F|
|[Mike, Mary, Will…|[Python, VB]|   OH|     M|
+--------------------+------------+-----+------+
```

[89]:
```
df.filter(df.name.lastname == "Williams").explain()
```

```
== Physical Plan ==
*(1) Filter (isnotnull(name#2230) AND (name#2230.lastname = Williams))
+- *(1) Scan ExistingRDD[name#2230,languages#2231,state#2232,gender#2233]
```

[91]:
```
df.filter(array_contains(df.languages,"Java")).explain()
```

```
== Physical Plan ==
*(1) Filter array_contains(languages#2231, Java)
+- *(1) Scan ExistingRDD[name#2230,languages#2231,state#2232,gender#2233]
```

[92]:
```
df.filter((df.state == "OH") & (df.gender == "M")).explain()
```

```
== Physical Plan ==
*(1) Filter (((isnotnull(state#2232) AND isnotnull(gender#2233)) AND (state#2232
= OH)) AND (gender#2233 = M))
+- *(1) Scan ExistingRDD[name#2230,languages#2231,state#2232,gender#2233]
```

[93]:
```
# Przyklad 2
```

[94]:
```
schema = StructType()
schema.add("fname",StringType())
schema.add("lname", StringType())
schema.add("courses", StringType())
schema.add("grade", IntegerType())
schema.add("year", IntegerType())
```

[94]:
```
StructType(List(StructField(fname,StringType,true),StructField(lname,StringType,
true),StructField(courses,StringType,true),StructField(grade,IntegerType,true),S
tructField(year,IntegerType,true)))
```

[97]:
```
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[4]"))
session = SparkSession(sc)
```

[99]:
```
data = session.read.format("csv").options(header='True', delimiter='|').
↪schema(schema).load("/home/spark/files/dane1.csv")
```

[100]:
```
data.where("fname == 'Weronika'").show()
data.where(data.fname == "Weronika").show()
data.where(col("fname") == "Weronika").show()
data.select("lname").show()
data.select("lname","fname").show()
data.groupBy("grade","year").count().show()
data.groupBy("grade","year").count().orderBy("grade","year").show()
```

```
+--------+----------+-----------+-----+----+
|   fname|     lname|    courses|grade|year|
+--------+----------+-----------+-----+----+
|Weronika|    Kapłon|     FIZYKA|    1|   5|
|Weronika|    Stanek|     FIZYKA|    1|   5|
|Weronika| Wiszyńska|INFORMATYKA|    1|   3|
|Weronika| Mrozińska|     FIZYKA|    2|   2|
|Weronika|  Szewczyk|     FIZMED|    1|   3|
|Weronika|   Szpytma|INFORMATYKA|    1|   7|
|Weronika|Schabowicz|INFORMATYKA|    1|   7|
|Weronika| Pastuszka|     FIZMED|    1|   7|
|Weronika|  Miszczak|INFORMATYKA|    1|   3|
|Weronika|     Tracz|     FIZMED|    1|   5|
|Weronika|    Ciurej|INFORMATYKA|    1|   3|
+--------+----------+-----------+-----+----+

+--------+----------+-----------+-----+----+
|   fname|     lname|    courses|grade|year|
+--------+----------+-----------+-----+----+
|Weronika|    Kapłon|     FIZYKA|    1|   5|
|Weronika|    Stanek|     FIZYKA|    1|   5|
|Weronika| Wiszyńska|INFORMATYKA|    1|   3|
|Weronika| Mrozińska|     FIZYKA|    2|   2|
|Weronika|  Szewczyk|     FIZMED|    1|   3|
|Weronika|   Szpytma|INFORMATYKA|    1|   7|
|Weronika|Schabowicz|INFORMATYKA|    1|   7|
|Weronika| Pastuszka|     FIZMED|    1|   7|
|Weronika|  Miszczak|INFORMATYKA|    1|   3|
|Weronika|     Tracz|     FIZMED|    1|   5|
|Weronika|    Ciurej|INFORMATYKA|    1|   3|
+--------+----------+-----------+-----+----+

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

```
|   fname|     lname|    courses|grade|year|
+--------+----------+-----------+-----+----+
|Weronika|    Kapłon|     FIZYKA|    1|   5|
|Weronika|    Stanek|     FIZYKA|    1|   5|
|Weronika| Wiszyńska|INFORMATYKA|    1|   3|
|Weronika| Mrozińska|     FIZYKA|    2|   2|
|Weronika|  Szewczyk|     FIZMED|    1|   3|
|Weronika|   Szpytma|INFORMATYKA|    1|   7|
|Weronika|Schabowicz|INFORMATYKA|    1|   7|
|Weronika| Pastuszka|     FIZMED|    1|   7|
|Weronika|  Miszczak|INFORMATYKA|    1|   3|
|Weronika|     Tracz|     FIZMED|    1|   5|
|Weronika|    Ciurej|INFORMATYKA|    1|   3|
+--------+----------+-----------+-----+----+

+-----------+
|      lname|
+-----------+
|     Kapłon|
|     Snażyk|
|  Brockhuis|
|     Kawala|
|     Piętka|
|   Pietruch|
|   Kukiełka|
|       Król|
| Kramarczyk|
|     Popiel|
|    Tomczyk|
|      Mazur|
|       Kruk|
|      Gałat|
|      Śledź|
|      Bizoń|
|  Tyczyński|
|  Zborowska|
|     Damian|
|Piwowarczyk|
+-----------+
only showing top 20 rows

+-----------+----------+
|      lname|     fname|
+-----------+----------+
|     Kapłon|  Weronika|
|     Snażyk|   Izabela|
|  Brockhuis|       Leo|
|     Kawala|    Alicja|
```

```
|     Piętka|   Bartosz|
|   Pietruch|     Dawid|
|   Kukiełka|     Piotr|
|       Król| Stanisław|
| Kramarczyk|Franciszek|
|     Popiel|Aleksandra|
|    Tomczyk|     Kamil|
|      Mazur|    Hubert|
|       Kruk| Tymoteusz|
|      Gałat|    Robert|
|      Śledź|    Patryk|
|      Bizoń|   Jadwiga|
|  Tyczyński|     Rafał|
|  Zborowska|    Joanna|
|     Damian|     Rafał|
|Piwowarczyk|    Michał|
+-----------+----------+
only showing top 20 rows

+-----+----+-----+
|grade|year|count|
+-----+----+-----+
|    2|   2|   95|
|    1|   7|  137|
| null|null|    1|
|    1|   3|  203|
|    1|   5|  139|
+-----+----+-----+

+-----+----+-----+
|grade|year|count|
+-----+----+-----+
| null|null|    1|
|    1|   3|  203|
|    1|   5|  139|
|    1|   7|  137|
|    2|   2|   95|
+-----+----+-----+
```

[101]:
```
data.createGlobalTempView("lista")
spark.sql("SELECT * from global_temp.lista").show()
spark.sql("SELECT grade, year, count(*) from global_temp.lista group by grade,␣
 ↪year order by grade, year").show()
```

```
+---------+----------+----------+-----+----+
|    fname|     lname|   courses|grade|year|
+---------+----------+----------+-----+----+
```

```
|  Weronika|     Kapłon|     FIZYKA|    1|   5|
|   Izabela|     Snażyk|INFORMATYKA|    1|   5|
|       Leo|  Brockhuis|     FIZYKA|    2|   2|
|    Alicja|     Kawala|     FIZYKA|    1|   7|
|   Bartosz|     Piętka|INFORMATYKA|    1|   3|
|     Dawid|   Pietruch|     FIZYKA|    1|   5|
|     Piotr|   Kukiełka|     FIZYKA|    2|   2|
| Stanisław|       Król|     FIZYKA|    2|   2|
|Franciszek| Kramarczyk|INFORMATYKA|    1|   5|
|Aleksandra|     Popiel|     FIZMED|    2|   2|
|     Kamil|    Tomczyk|INFORMATYKA|    1|   7|
|    Hubert|      Mazur|INFORMATYKA|    1|   5|
| Tymoteusz|       Kruk|INFORMATYKA|    2|   2|
|    Robert|      Gałat|INFORMATYKA|    2|   2|
|    Patryk|      Śledź|INFORMATYKA|    1|   3|
|   Jadwiga|      Bizoń|     FIZMED|    1|   3|
|     Rafał|  Tyczyński|     FIZMED|    1|   7|
|    Joanna|  Zborowska|     FIZMED|    1|   7|
|     Rafał|     Damian|     FIZYKA|    1|   7|
|    Michał|Piwowarczyk|INFORMATYKA|    1|   7|
+----------+-----------+-----------+-----+----+
only showing top 20 rows

+-----+----+--------+
|grade|year|count(1)|
+-----+----+--------+
| null|null|       1|
|    1|   3|     203|
|    1|   5|     139|
|    1|   7|     137|
|    2|   2|      95|
+-----+----+--------+
```

[102]:
```
#ZAD1
```

[113]:
```
spark.sql("SELECT grade, year, count(*) AS liczba_studentow from global_temp.
 ↪lista WHERE grade = '1' GROUP BY grade, year ORDER BY liczba_studentow␣
 ↪DESC").show()
```

```
+-----+----+---------------+
|grade|year|liczba_studentow|
+-----+----+---------------+
|    1|   3|            203|
|    1|   5|            139|
|    1|   7|            137|
+-----+----+---------------+
```

[104]:
```
#ZAD2
```

[114]:
```
spark.sql("SELECT COUNT(lname) AS liczba_studentow FROM global_temp.lista").
↪show()
```

```
+----------------+
|liczba_studentow|
+----------------+
|             575|
+----------------+
```

[106]:
```
#ZAD3
```

[116]:
```
spark.sql("SELECT COUNT(DISTINCT fname) AS liczba_imion FROM global_temp.
↪lista").show()
```

```
+------------+
|liczba_imion|
+------------+
|         136|
+------------+
```