In [6]:
```
import pandas as pd
import numpy as np
pd.set_option('display.max.columns', 100)
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
data = pd.read_csv('2014_ebola.csv')
data.head()
```

Out[7]:

	Country	Month	Year	Lat	Lon	Value
0	Guinea	3	14	9.95	-9.7	122
1	Guinea	4	14	9.95	-9.7	224
2	Guinea	5	14	9.95	-9.7	291
3	Guinea	6	14	9.95	-9.7	413
4	Guinea	7	14	9.95	-9.7	460

In [8]:
```
#1. How many countries are represented in this dataset?
data['Country'].value_counts()
```

Out[8]:
```
Guinea          10
Liberia          9
Sierra Leone     8
Mali             2
Senegal          2
Name: Country, dtype: int64
```

In [9]:
```
#2. What is the average value of Guinea?
Guinea_data = data[data['Country'] == 'Guinea']
```

In [10]:
```
Guinea_data['Value'].mean()
```

Out[10]: 596.1

In [11]:
```
#alt
data.loc[data['Country'] == 'Guinea', 'Value'].mean()
```

Out[11]: 596.1

In [15]:
```
#3. What is the proportion of German citizens (native-country featu
re)?
float((data['Country'] == 'Guinea').sum()) / data.shape[0]
```

Out[15]:
```
0.3225806451612903
```

In [26]:
```
#4-5. What are mean value and standard deviation of the age
#of those who recieve more than 50K per year (salary feature) and t
hose who receive less than 50K per year?
Guinea = data.loc[data['Country'] == 'Guinea', 'Value']
Senegal = data.loc[data['Country'] == 'Senegal', 'Value']

print("The average Value of second half of year: {0}. The average V
alue of first half of year - {1} ".format(
    round(Guinea.mean()),
    round(Senegal.mean())
))
```

```
The average Value of second half of year: 596.0. The average Value
of first half of year - 2.0
```

In [27]:
```
#6. Is it true that people who receive more than 50k have at least
high school education?
#(education - Bachelors, Prof-school, Assoc-acdm, Assoc-voc, Master
s or Doctorate feature)
data.loc[data['Country'] == 'Liberia', 'Lat'].unique()
```

Out[27]:
```
array([6.43])
```

In [28]:
```
#7. Display statistics of age for each race (race feature) and each
gender.
#Use groupby() and describe(). Find the maximum age of men of Amer-
Indian-Eskimo race.
#data.groupby(['race', 'sex']).describe()
for (Country, Month), sub_df in data.groupby(['Country', 'Month']):
    print('Country: {0}, Month {1}'.format(Country, Month))
    print(sub_df['Month'].describe())
```

```
Country: Guinea, Month 3
count    1.0
mean     3.0
std      NaN
min      3.0
25%      3.0
50%      3.0
75%      3.0
max      3.0
Name: Month, dtype: float64
Country: Guinea, Month 4
count    1.0
mean     4.0
std      NaN
```

```
min      4.0
25%      4.0
50%      4.0
75%      4.0
max      4.0
Name: Month, dtype: float64
Country: Guinea, Month 5
count    1.0
mean     5.0
std      NaN
min      5.0
25%      5.0
50%      5.0
75%      5.0
max      5.0
Name: Month, dtype: float64
Country: Guinea, Month 6
count    1.0
mean     6.0
std      NaN
min      6.0
25%      6.0
50%      6.0
75%      6.0
max      6.0
Name: Month, dtype: float64
Country: Guinea, Month 7
count    1.0
mean     7.0
std      NaN
min      7.0
25%      7.0
50%      7.0
75%      7.0
max      7.0
Name: Month, dtype: float64
Country: Guinea, Month 8
count    1.0
mean     8.0
std      NaN
min      8.0
25%      8.0
50%      8.0
75%      8.0
max      8.0
Name: Month, dtype: float64
Country: Guinea, Month 9
count    1.0
mean     9.0
std      NaN
min      9.0
25%      9.0
50%      9.0
```

```
75%      9.0
max      9.0
Name: Month, dtype: float64
Country: Guinea, Month 10
count     1.0
mean     10.0
std       NaN
min      10.0
25%      10.0
50%      10.0
75%      10.0
max      10.0
Name: Month, dtype: float64
Country: Guinea, Month 11
count     1.0
mean     11.0
std       NaN
min      11.0
25%      11.0
50%      11.0
75%      11.0
max      11.0
Name: Month, dtype: float64
Country: Guinea, Month 12
count     1.0
mean     12.0
std       NaN
min      12.0
25%      12.0
50%      12.0
75%      12.0
max      12.0
Name: Month, dtype: float64
Country: Liberia, Month 4
count    1.0
mean     4.0
std      NaN
min      4.0
25%      4.0
50%      4.0
75%      4.0
max      4.0
Name: Month, dtype: float64
Country: Liberia, Month 5
count    1.0
mean     5.0
std      NaN
min      5.0
25%      5.0
50%      5.0
75%      5.0
max      5.0
Name: Month, dtype: float64
```

```
Country: Liberia, Month 6
count    1.0
mean     6.0
std      NaN
min      6.0
25%      6.0
50%      6.0
75%      6.0
max      6.0
Name: Month, dtype: float64
Country: Liberia, Month 7
count    1.0
mean     7.0
std      NaN
min      7.0
25%      7.0
50%      7.0
75%      7.0
max      7.0
Name: Month, dtype: float64
Country: Liberia, Month 8
count    1.0
mean     8.0
std      NaN
min      8.0
25%      8.0
50%      8.0
75%      8.0
max      8.0
Name: Month, dtype: float64
Country: Liberia, Month 9
count    1.0
mean     9.0
std      NaN
min      9.0
25%      9.0
50%      9.0
75%      9.0
max      9.0
Name: Month, dtype: float64
Country: Liberia, Month 10
count     1.0
mean     10.0
std       NaN
min      10.0
25%      10.0
50%      10.0
75%      10.0
max      10.0
Name: Month, dtype: float64
Country: Liberia, Month 11
count     1.0
mean     11.0
```

```
std       NaN
min      11.0
25%      11.0
50%      11.0
75%      11.0
max      11.0
Name: Month, dtype: float64
Country: Liberia, Month 12
count     1.0
mean     12.0
std       NaN
min      12.0
25%      12.0
50%      12.0
75%      12.0
max      12.0
Name: Month, dtype: float64
Country: Mali, Month 10
count     1.0
mean     10.0
std       NaN
min      10.0
25%      10.0
50%      10.0
75%      10.0
max      10.0
Name: Month, dtype: float64
Country: Mali, Month 11
count     1.0
mean     11.0
std       NaN
min      11.0
25%      11.0
50%      11.0
75%      11.0
max      11.0
Name: Month, dtype: float64
Country: Senegal, Month 8
count    1.0
mean     8.0
std      NaN
min      8.0
25%      8.0
50%      8.0
75%      8.0
max      8.0
Name: Month, dtype: float64
Country: Senegal, Month 9
count    1.0
mean     9.0
std      NaN
min      9.0
25%      9.0
```

```
50%      9.0
75%      9.0
max      9.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 5
count    1.0
mean     5.0
std      NaN
min      5.0
25%      5.0
50%      5.0
75%      5.0
max      5.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 6
count    1.0
mean     6.0
std      NaN
min      6.0
25%      6.0
50%      6.0
75%      6.0
max      6.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 7
count    1.0
mean     7.0
std      NaN
min      7.0
25%      7.0
50%      7.0
75%      7.0
max      7.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 8
count    1.0
mean     8.0
std      NaN
min      8.0
25%      8.0
50%      8.0
75%      8.0
max      8.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 9
count    1.0
mean     9.0
std      NaN
min      9.0
25%      9.0
50%      9.0
75%      9.0
max      9.0
```

```
Name: Month, dtype: float64
Country: Sierra Leone, Month 10
count     1.0
mean     10.0
std       NaN
min      10.0
25%      10.0
50%      10.0
75%      10.0
max      10.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 11
count     1.0
mean     11.0
std       NaN
min      11.0
25%      11.0
50%      11.0
75%      11.0
max      11.0
Name: Month, dtype: float64
Country: Sierra Leone, Month 12
count     1.0
mean     12.0
std       NaN
min      12.0
25%      12.0
50%      12.0
75%      12.0
max      12.0
Name: Month, dtype: float64
```

1. Among whom the proportion of those who earn a lot(>50K) is more: among married or single men (marital-status feature)? Consider married those who have a marital-status starting with Married (Married-civ-spouse, Married-spouse-absent or Married-AF-spouse), the rest are considered bachelors.

In [30]:
```
data.loc[
    (data['Country'] == 'Guinea') &
    (data['Month'].isin(['1','2','3','4','5','6','7','8','9','10','
11','12'])), 'Value'
].value_counts()
```

Out[30]:
```
1023    1
1022    1
413     1
460     1
122     1
867     1
771     1
291     1
768     1
224     1
Name: Value, dtype: int64
```

In [37]:
```
data.loc[(data['Country'] == 'Guinea') & (data['Month'] == 6), 'Val
ue'].value_counts()
```

Out[37]:
```
413    1
Name: Value, dtype: int64
```

In [38]:
```
data['Month'].value_counts()
```

Out[38]:
```
11    4
10    4
9     4
8     4
12    3
7     3
6     3
5     3
4     2
3     1
Name: Month, dtype: int64
```

1. What is the maximum number of hours a person works per week (hours-per-week feature)? How many people work such a number of hours and what is the percentage of those who earn a lot among them?

In [43]:
```
maximum = data['Lat'].max()
print("Max Lat - {0}".format(maximum))

total = data[data['Lat'] == maximum].shape[0]
print("Total number Lat {0}".format(total))

percentage = float(data[(data['Lat'] == maximum)
                 & (data['Value'] == '>1000')].shape[0]) / total
print("Percentage {0}%".format(int(100 * percentage)))
```

```
Max Lat - 17.57
Total number Lat 2
Percentage 0%
```

1. Count the average time of work (hours-per-week) those who earning a little and a lot (salary) for each country (native-country).

In [44]:
```
for (Country, Value), sub_df in data.groupby(['Country', 'Value']):
    print(Country, Value, round(sub_df['Lat'].mean(), 2))
```

```
Guinea 122 9.95
Guinea 224 9.95
Guinea 291 9.95
Guinea 413 9.95
Guinea 460 9.95
Guinea 768 9.95
Guinea 771 9.95
Guinea 867 9.95
Guinea 1022 9.95
Guinea 1023 9.95
Liberia 13 6.43
Liberia 35 6.43
Liberia 107 6.43
Liberia 329 6.43
Liberia 1395 6.43
Liberia 2765 6.43
Liberia 3362 6.43
Liberia 3499 6.43
Liberia 3567 6.43
Mali 1 17.57
Mali 8 17.57
Senegal 1 14.5
Senegal 3 14.5
Sierra Leone 50 8.46
Sierra Leone 239 8.46
Sierra Leone 533 8.46
Sierra Leone 1216 8.46
Sierra Leone 1856 8.46
Sierra Leone 1930 8.46
Sierra Leone 1934 8.46
Sierra Leone 1940 8.46
```

In [45]:
```
pd.crosstab(data['Country'], data['Value'],
            values=data['Lat'], aggfunc=np.mean).T
```

Out[45]:

Country	Guinea	Liberia	Mali	Senegal	Sierra Leone
Value					
1	NaN	NaN	17.57	14.5	NaN
3	NaN	NaN	NaN	14.5	NaN
8	NaN	NaN	17.57	NaN	NaN
13	NaN	6.43	NaN	NaN	NaN
35	NaN	6.43	NaN	NaN	NaN
50	NaN	NaN	NaN	NaN	8.46
107	NaN	6.43	NaN	NaN	NaN
122	9.95	NaN	NaN	NaN	NaN
224	9.95	NaN	NaN	NaN	NaN
239	NaN	NaN	NaN	NaN	8.46
291	9.95	NaN	NaN	NaN	NaN
329	NaN	6.43	NaN	NaN	NaN
413	9.95	NaN	NaN	NaN	NaN
460	9.95	NaN	NaN	NaN	NaN
533	NaN	NaN	NaN	NaN	8.46
768	9.95	NaN	NaN	NaN	NaN
771	9.95	NaN	NaN	NaN	NaN
867	9.95	NaN	NaN	NaN	NaN
1022	9.95	NaN	NaN	NaN	NaN
1023	9.95	NaN	NaN	NaN	NaN
1216	NaN	NaN	NaN	NaN	8.46
1395	NaN	6.43	NaN	NaN	NaN
1856	NaN	NaN	NaN	NaN	8.46
1930	NaN	NaN	NaN	NaN	8.46
1934	NaN	NaN	NaN	NaN	8.46
1940	NaN	NaN	NaN	NaN	8.46
2765	NaN	6.43	NaN	NaN	NaN
3362	NaN	6.43	NaN	NaN	NaN
3499	NaN	6.43	NaN	NaN	NaN
3567	NaN	6.43	NaN	NaN	NaN

PART 2

In [66]:
```
user_usage = pd.read_csv("user_usage.csv")
user_device = pd.read_csv("user_device.csv")
android_devices = pd.read_csv("android_devices.csv")
```

```
---------------------------------------------------------------------
---------
FileNotFoundError                         Traceback (most recent c
all last)
<ipython-input-66-3d199d8c6606> in <module>
----> 1 user_usage = pd.read_csv("user_usage.csv")
      2 user_device = pd.read_csv("user_device.csv")
      3 android_devices = pd.read_csv("android_devices.csv")

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in par
ser_f(filepath_or_buffer, sep, delimiter, header, names, index_col
, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, conve
rters, true_values, false_values, skipinitialspace, skiprows, skip
footer, nrows, na_values, keep_default_na, na_filter, verbose, ski
p_blank_lines, parse_dates, infer_datetime_format, keep_date_col,
date_parser, dayfirst, cache_dates, iterator, chunksize, compressi
on, thousands, decimal, lineterminator, quotechar, quoting, double
quote, escapechar, comment, encoding, dialect, error_bad_lines, wa
rn_bad_lines, delim_whitespace, low_memory, memory_map, float_prec
ision)
    674         )
    675
--> 676         return _read(filepath_or_buffer, kwds)
    677
    678     parser_f.__name__ = name

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in _re
ad(filepath_or_buffer, kwds)
    446
    447     # Create the parser.
--> 448     parser = TextFileReader(fp_or_buf, **kwds)
    449
    450     if chunksize or iterator:

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in __i
nit__(self, f, engine, **kwds)
    878             self.options["has_index_names"] = kwds["has_in
dex_names"]
    879
--> 880         self._make_engine(self.engine)
    881
    882     def close(self):

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in _ma
```

```
ke_engine(self, engine)
   1112     def _make_engine(self, engine="c"):
   1113         if engine == "c":
-> 1114             self._engine = CParserWrapper(self.f, **self.o
ptions)
   1115         else:
   1116             if engine == "python":

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in __i
nit__(self, src, **kwds)
   1889         kwds["usecols"] = self.usecols
   1890
-> 1891         self._reader = parsers.TextReader(src, **kwds)
   1892         self.unnamed_cols = self._reader.unnamed_cols
   1893

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cini
t__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup
_parser_source()

FileNotFoundError: [Errno 2] File user_usage.csv does not exist: '
user_usage.csv'
```

In [61]:
```
user_usage.head()
```

Out[61]:

	num	lat	long	depth	mag	stations
0	1	-20.42	181.62	562.0	04.Aug	41
1	2	-20.62	181.03	650.0	04.Feb	15
2	3	-26.00	184.10	42.0	05.Apr	43
3	4	-17.97	181.66	626.0	04.Jan	19
4	5	-20.42	181.96	649.0	4	11

In [62]:
```
user_device.head()
```

Out[62]:

	num,"agegp","alcgp","tobgp","ncases","ncontrols"
0	1,"25-34","0-39g/day","0-9g/day",0,40
1	2,"25-34","0-39g/day","10-19",0,10
2	3,"25-34","0-39g/day","20-29",0,6
3	4,"25-34","0-39g/day","30+",0,5
4	5,"25-34","40-79","0-9g/day",0,27

In [65]:
```
merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id')
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
<ipython-input-65-29ccc59f735e> in <module>
----> 1 merged = pd.merge(user_usage,
      2                   user_device[['use_id', 'device', 'platfo
rm']],
      3                   on='use_id')

NameError: name 'user_usage' is not defined
```

In [28]:
```
merged.head()
```

Out[28]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	device	platfo
0	21.97	4.82	1557.33	22787	GT-I9505	andr
1	1710.08	136.88	7267.55	22788	SM-G930F	andr
2	1710.08	136.88	7267.55	22789	SM-G930F	andr
3	94.46	35.17	519.12	22790	D2303	andr
4	71.59	79.26	1557.33	22792	SM-G361F	andr

In [29]:
```
print("user_usage dimensions: {}".format(user_usage.shape))
print("user_device dimensions: {}".format(user_device[['use_id', 'p
latform', 'device']].shape))
```

```
user_usage dimensions: (240, 4)
user_device dimensions: (272, 3)
```

In [30]:
```
user_usage['use_id'].isin(user_device['use_id']).value_counts()
```

Out[30]:
```
True     159
False     81
Name: use_id, dtype: int64
```

In [34]:
```
merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id',
                   how='left')
print("user_usage dimensions: {}".format(user_usage.shape))
print("merged dimensions: {}".format(merged.shape))
print("Missing values: {}".format(merged['device'].isnull().sum()))
```

```
user_usage dimensions: (240, 4)
merged dimensions: (240, 6)
Missing values: 81
```

In [33]:
```
merged.tail()
```

Out[33]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	device	plat
235	260.66	68.44	896.96	25008	NaN	
236	97.12	36.50	2815.00	25040	NaN	
237	355.93	12.37	6828.09	25046	NaN	
238	632.06	120.46	1453.16	25058	NaN	
239	488.70	906.92	3089.85	25220	NaN	

In [35]:
```
merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id',
                  how='right')
print("user_device dimensions: {}".format(user_device.shape))
print("merged dimensions: {}".format(merged.shape))
print("Missing values in monthly_mb: {}".format(
        merged['monthly_mb'].isnull().sum()))
print("Missing values in platform: {}".format(
        merged['platform'].isnull().sum()))
```

```
user_device dimensions: (272, 6)
merged dimensions: (272, 6)
Missing values in monthly_mb: 113
Missing values in platform: 0
```

In [37]:
```
merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id',
                  how='outer',
                  indicator=True)

print("Rows in outer merge: {}".format(merged.shape))

print("No missing values: {}".format(
    (merged.apply(lambda x: x.isnull().sum(), axis=1) == 0).sum()))
```

```
Rows in outer merge: (353, 7)
No missing values: 159
```

In [44]:
```
# First, add the platform and device to the user usage.
merged = pd.merge(user_usage,
                  user_device[['use_id', 'platform', 'device']],
                  on='use_id',
                  how='left')

# Now, based on the "device" column in result, match the "Model" co
lumn in devices.
android_devices.rename(columns={"Retail Branding": "manufacturer"},
                       inplace=True)
merged = pd.merge(result,
                  android_devices[['manufacturer', 'Model']],
                  left_on='device',
                  right_on='Model',
                  how='left')

merged.head()
```

Out[44]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	devi
0	21.97	4.82	1557.33	22787	android	C I95
1	1710.08	136.88	7267.55	22788	android	S G93
2	1710.08	136.88	7267.55	22789	android	S G93
3	94.46	35.17	519.12	22790	android	D23
4	71.59	79.26	1557.33	22792	android	S G36

In [46]:
```
android_devices[android_devices.Model == 'SM-G930F']
```

Out[46]:

	manufacturer	Marketing Name	Device	Model
10381	Samsung	Galaxy S7	herolte	SM-G930F

In [47]:
```
android_devices[android_devices.Device.str.startswith('GT')]
```

Out[47]:

	manufacturer	Marketing Name	Device	Model
1095	Bitmore	GTAB700	GTAB700	NID_7010
1096	Bitmore	GTAB900	GTAB900	S952
2402	Grundig	GTB1050	GTB1050	GTB 1050
2403	Grundig	GTB850	GTB850	GTB 850
2404	Grundig	TC69CA2	GTB801	GTB 801
...	...	...	...	...
10821	Samsung	Galaxy Y Pro	GT-B5510L	GT-B5510L
10822	Samsung	Galaxy Y Pro Duos	GT-B5512	GT-B5512
10823	Samsung	Galaxy Y Pro Duos	GT-B5512B	GT-B5512B
10824	Samsung	Galaxy Y TV	GT-S5367	GT-S5367
10979	Sharp	AQUOS SERIE mini SHV38	GTQ	SHV38

164 rows × 4 columns

In [41]:
```
merged.head()
```

Out[41]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	devi
0	21.97	4.82	1557.33	22787	android	C I95
1	1710.08	136.88	7267.55	22788	android	S G93
2	1710.08	136.88	7267.55	22789	android	S G93
3	94.46	35.17	519.12	22790	android	D23
4	71.59	79.26	1557.33	22792	android	S G36

In [48]:
```
merged.groupby("manufacturer").agg({
        "outgoing_mins_per_month": "mean",
        "outgoing_sms_per_month": "mean",
        "monthly_mb": "mean",
        "use_id": "count"
    })
```

Out[48]:

manufacturer	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
HTC	299.842955	93.059318	5144.077955	44
Huawei	81.526667	9.500000	1561.226667	3
LGE	111.530000	12.760000	1557.330000	2
Lava	60.650000	261.900000	12458.670000	2
Lenovo	215.920000	12.930000	1557.330000	2
Motorola	95.127500	65.666250	3946.500000	16
OnePlus	354.855000	48.330000	6575.410000	6
Samsung	191.010093	92.390463	4017.318889	108
Sony	177.315625	40.176250	3212.000625	16
Vodafone	42.750000	46.830000	5191.120000	1
ZTE	42.750000	46.830000	5191.120000	1

Pandasql

In [ ]:
```
import pandasql as ps
```

In [22]:
```
# pandasql code
def lj_pandasql(user_usage, user_device):
    join_query = '''
       SELECT user_usage.outgoing_mins_per_month,
              user_usage.outgoing_sms_per_month,
              user_usage.monthly_mb,
              user_device.use_id,
              user_device.device,
              user_device.platform

        FROM user_usage
        LEFT JOIN user_device
        ON user_usage.use_id = user_device.use_id;
    '''
    return ps.sqldf(join_query, locals())
lj_pandasql(user_usage, user_device).tail()
```

Out[22]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	device	plat
235	260.66	68.44	896.96	NaN	None	
236	97.12	36.50	2815.00	NaN	None	
237	355.93	12.37	6828.09	NaN	None	
238	632.06	120.46	1453.16	NaN	None	
239	488.70	906.92	3089.85	NaN	None	

Агрегирование

pandasql

In [112]:
```
from datetime import datetime
def aggr_pandasql(user_usage, user_device):
    aggreg_query = '''
       SELECT avg(user_usage.monthly_mb) as avg_monthly_mb,
              user_device.platform
        FROM user_usage
        LEFT JOIN user_device
        ON user_usage.use_id = user_device.use_id
        GROUP BY user_device.platform;
    '''
    return ps.sqldf(aggreg_query, locals())
aggr_pandasql(user_usage, user_device)
```

Out[112]:

	avg_monthly_mb	platform
0	2545.485062	None
1	4221.387834	android
2	961.155000	ios

In [51]:
```

```

Out[51]:
```
datetime.timedelta(microseconds=28319)
```

pandas

In [41]:
```
merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id',
                   how='left')
merged.groupby('platform').mean()
```

Out[41]:

platform	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
android	201.258535	85.354586	4221.387834	22922.350318
ios	366.060000	293.975000	961.155000	22920.500000

TIME

In [125]:
```
import time

def count_mean_time(func, params, N =5):
    total_time = 0
    for i in range(N):
        time1 = time.time()
        if len(params) == 1:
            tmp_df = func(params[0])
        elif len(params) == 2:
            tmp_df = func(params[0], params[1])
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N

lj_ps_mean = count_mean_time(lj_pandasql,
                             [user_usage, user_device], N=40)
lj_ps_mean
```

Out[125]: 0.014299607276916504

In [126]:
```
def pd_merge_group(user_usage, user_device):
    merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id',
                   how='left')
    res = merged.groupby('platform').mean()
    return res
pd_merge_group_mean = count_mean_time(pd_merge_group,
                             [user_usage, user_device], N=40)
pd_merge_group_mean
```

Out[126]: 0.007319676876068115

In [127]:
```
def pd_merge(user_usage, user_device):
    merged = pd.merge(user_usage,
                  user_device[['use_id', 'device', 'platform']],
                  on='use_id',
                   how='left')
    return merged
pd_merge_mean = count_mean_time(pd_merge,
                                [user_usage, user_device], N=40)
pd_merge_mean
```

Out[127]: 0.004346024990081787

In [128]:
```
aggr_ps_mean = count_mean_time(aggr_pandasql,
                             [user_usage, user_device], N=40)
aggr_ps_mean
```

Out[128]: 0.01447572112083435

In [129]:
```
merge_delta = lj_ps_mean - pd_merge_mean
merge_delta
```

Out[129]: 0.009953582286834718

In [130]:
```
aggr_delta = aggr_ps_mean - pd_merge_group_mean
aggr_delta
```

Out[130]: 0.0071560442447662345

Вывод: pandasql дольше работает