Изучение библиотек обработки данных.

In [1]:

```
#Часть 1
import numpy as np
import pandas as pd
```

In [2]:

```
data = pd.read_csv('adult.data.csv')
data.head()
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [3]:

```
#1. How many men and women (sex feature) are represented in this dataset?
data["sex"].value_counts()
```

Out[3]:

```
Male      21790
Female    10771
Name: sex, dtype: int64
```

In [4]:

```
#2. What is the average age (age feature) of women?
data.loc[data['sex'] == 'Female', 'age'].mean()
```

Out[4]:

```
36.85823043357163
```

In [5]:

```
#3. What is the proportion of German citizens (native-country feature)?
print("{0:%}".format(data[data["native-country"] == "Germany"]
                .shape[0] / data.shape[0]))
```

```
0.420749%
```

In [6]:

```
#4-5. What are the mean and standard deviation of age for those who earn more than 50K per year (salary feature)
#and those who earn less than 50K per year?

ages1 = data[data["salary"] == "<=50K"]["age"]
ages2 = data[data["salary"] ==  ">50K"]["age"]
print("<=50K: = {0} ± {1} years".format(ages1.mean(), ages1.std()))
print(" >50K: = {0} ± {1} years".format(ages2.mean(), ages2.std()))
```

```
<=50K: = 36.78373786407767 ± 14.020088490824813 years
```

```
<=50K: = 36.78373786407767 ± 14.020088490824813 years
 >50K: = 44.24984058155847 ± 10.51902771985177 years
```

In [7]:

```python
#6. Is it true that people who earn more than 50K have at least high school education?
#(education – Bachelors, Prof-school, Assoc-acdm, Assoc-voc, Masters or Doctorate feature)
high_educations = set(["Bachelors", "Prof-school", "Assoc-acdm",
                       "Assoc-voc", "Masters", "Doctorate"])
def high_educated(e):
    return e in high_educations

data[data["salary"] == ">50K"]["education"].map(high_educated).all()
```

Out[7]:

```
False
```

In [8]:

```python
#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.
for (race, sex), sub_df in data.groupby(['race', 'sex']):
    print("Race: {0}, sex: {1}".format(race, sex))
    print(sub_df['age'].describe())
```

```
Race: Amer-Indian-Eskimo, sex: Female
count    119.000000
mean      37.117647
std       13.114991
min       17.000000
25%       27.000000
50%       36.000000
75%       46.000000
max       80.000000
Name: age, dtype: float64
Race: Amer-Indian-Eskimo, sex: Male
count    192.000000
mean      37.208333
std       12.049563
min       17.000000
25%       28.000000
50%       35.000000
75%       45.000000
max       82.000000
Name: age, dtype: float64
Race: Asian-Pac-Islander, sex: Female
count    346.000000
mean      35.089595
std       12.300845
min       17.000000
25%       25.000000
50%       33.000000
75%       43.750000
max       75.000000
Name: age, dtype: float64
Race: Asian-Pac-Islander, sex: Male
count    693.000000
mean      39.073593
std       12.883944
min       18.000000
25%       29.000000
50%       37.000000
75%       46.000000
max       90.000000
Name: age, dtype: float64
Race: Black, sex: Female
count    1555.000000
mean       37.854019
std        12.637197
min        17.000000
25%        28.000000
50%        37.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: Black, sex: Male
count    1569.000000
mean       37.682600
std        12.882612
```

```
min        17.000000
25%        27.000000
50%        36.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: Other, sex: Female
count    109.000000
mean      31.678899
std       11.631599
min       17.000000
25%       23.000000
50%       29.000000
75%       39.000000
max       74.000000
Name: age, dtype: float64
Race: Other, sex: Male
count    162.000000
mean      34.654321
std       11.355531
min       17.000000
25%       26.000000
50%       32.000000
75%       42.000000
max       77.000000
Name: age, dtype: float64
Race: White, sex: Female
count    8642.000000
mean       36.811618
std        14.329093
min        17.000000
25%        25.000000
50%        35.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: White, sex: Male
count    19174.000000
mean        39.652498
std         13.436029
min         17.000000
25%         29.000000
50%         38.000000
75%         49.000000
max         90.000000
Name: age, dtype: float64
```

In [9]:

```
#8. Among whom is the proportion of those who earn a lot (>50K) greater: married or single men (marital-status feature)?
#Consider as 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.
data.loc[(data['sex'] == 'Male') &
    (data['marital-status'].isin(['Never-married',
                                  'Separated',
                                  'Divorced',
                                  'Widowed'])), 'salary'].value_counts()

Out[10]:

<=50K    7552
```

```
  File "<ipython-input-9-12be563d53cd>", line 10
    Out[10]:
       ^
SyntaxError: invalid syntax
```

In [ ]:

```
data.loc[(data['sex'] == 'Male') &
    (data['marital-status'].str.startswith('Married')), 'salary'].value_counts()
```

In [ ]:

```
data['marital-status'].value_counts()
```

In [ ]:

```
#9. 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?
max_load = data['hours-per-week'].max()
print("Max time - {0} hours./week.".format(max_load))

num_workaholics = data[data['hours-per-week'] == max_load].shape[0]
print("Total number of such hard workers {0}".format(num_workaholics))

rich_share = float(data[(data['hours-per-week'] == max_load)
               & (data['salary'] == '>50K')].shape[0]) / num_workaholics
print("Percentage of rich among them {0}%".format(int(100 * rich_share)))
```

In [ ]:

```
#10. Count the average time of work (hours-per-week) those who earning a little and a lot (salary) for each country
#(native-country).
pd.crosstab(data['native-country'], data['salary'],
          values=data['hours-per-week'], aggfunc=np.mean).T
```

In [10]:

```
#Часть 2
!pip install pandasql
```

```
Requirement already satisfied: pandasql in c:\users\dovlat\anaconda3\lib\site-packages (0.7.3)
Requirement already satisfied: sqlalchemy in c:\users\dovlat\anaconda3\lib\site-packages (from pandasql) (1.3.1)
Requirement already satisfied: pandas in c:\users\dovlat\anaconda3\lib\site-packages (from pandasql) (0.24.2)
Requirement already satisfied: numpy in c:\users\dovlat\anaconda3\lib\site-packages (from pandasql) (1.16.2)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\dovlat\anaconda3\lib\site-packages (from pandas->pandasql) (2.8.0)
Requirement already satisfied: pytz>=2011k in c:\users\dovlat\anaconda3\lib\site-packages (from pandas->pandasql) (2018.9)
Requirement already satisfied: six>=1.5 in c:\users\dovlat\anaconda3\lib\site-packages (from python-dateutil>=2.5.0->pandas->pandasql) (1.12.0)
```

In [11]:

```
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

In [12]:

```
user_usage = pd.read_csv('user_usage.csv')
```

In [13]:

```
user_device = pd.read_csv('user_device.csv')
```

In [14]:

```
user_usage.head()
```

Out[14]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	21.97	4.82	1557.33	22787
1	1710.08	136.88	7267.55	22788
2	1710.08	136.88	7267.55	22789
3	94.46	35.17	519.12	22790
4	71.59	79.26	1557.33	22792

In [15]:

```
user_usage.dtypes
```

Out[15]:

```
outgoing_mins_per_month    float64
outgoing_sms_per_month     float64
monthly_mb                 float64
use_id                       int64
dtype: object
```

In [16]:

```
user_device.head()
```

Out[16]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1

In [17]:

```
user_device.dtypes
```

Out[17]:

```
use_id               int64
user_id              int64
platform            object
platform_version    float64
device              object
use_type_id          int64
dtype: object
```

In [18]:

```
user_device.merge(user_usage).head()
```

Out[18]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb
0	22787	12921	android	4.3	GT-I9505	1	21.97	4.82	1557.33
1	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55
2	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55
3	22790	29592	android	5.1	D2303	1	94.46	35.17	519.12
4	22792	28217	android	5.1	SM-G361F	1	71.59	79.26	1557.33

In [19]:

```
%%timeit
user_device.merge(user_usage).head()
```

```
15.7 ms ± 978 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [20]:

```
pysqldf("""SELECT  u.outgoing_mins_per_month, u.outgoing_sms_per_month, u.monthly_mb, u.use_id, d.user_id,d.platform,
d.platform_version, d.device, d.use_type_id
FROM user_usage AS u JOIN user_device AS d
ON u.use_id = d.use_id
""").head()
```

Out[20]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	user_id	platform	platform_version	device	use_type_id
0	21.97	4.82	1557.33	22787	12921	android	4.3	GT-I9505	1
1	1710.08	136.88	7267.55	22788	28714	android	6.0	SM-G930F	1
2	1710.08	136.88	7267.55	22789	28714	android	6.0	SM-G930F	1
3	94.46	35.17	519.12	22790	29592	android	5.1	D2303	1

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	user_id	platform	platform_version	device	use_type_id
4	71.59	79.26	1557.33	22792	28217	android	5.1	SM-G361F	1

In [21]:

```
%%timeit
pysqldf("""SELECT  u.outgoing_mins_per_month, u.outgoing_sms_per_month, u.monthly_mb, u.use_id, d.user_id,d.platform,
d.platform_version, d.device, d.use_type_id
FROM user_usage AS u JOIN user_device AS d
ON u.use_id = d.use_id
""")
```

```
59.6 ms ± 8.11 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [22]:

```
user_usage.groupby("use_id")["monthly_mb"].mean().head()
```

Out[22]:

```
use_id
22787    1557.33
22788    7267.55
22789    7267.55
22790     519.12
22792    1557.33
Name: monthly_mb, dtype: float64
```

In [23]:

```
%%timeit
user_usage.groupby("use_id")["monthly_mb"].mean()
```

```
2.84 ms ± 273 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```