Задание: Часть 1. Выполните первое демонстрационное задание "demo assignment" под названием "Exploratory data analysis with Pandas" со страницы курса https://mlcourse.ai/assignments (https://mlcourse.ai/assignments) Условие задания - https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_flush_cache=true (https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_flush_cache=true) Часть 2. Выполните следующие запросы с использованием двух различных библиотек - Pandas и PandaSQL: один произвольный запрос на соединение двух наборов данных один произвольный запрос на группировку набора данных с использованием функций агрегирования Сравните время выполнения каждого запроса в Pandas и PandaSQL.

In [32]:
```
import numpy as np #библиотека для работы с многомерными массивами данных и математическими операциями над ними
import pandas as pd #библиотека для анализа и обработки данных
pd.set_option('display.max.columns', 100)
import matplotlib.pyplot as plt #простое рисование графиков
import seaborn as sns #удобные дефолтные настройки графиков из matpotlib
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [33]:
```
data = pd.read_csv('./data/adult.data.csv')
data.head()
```

Out[33]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black

In [34]:
```
# 1) Сколько мужчин и женщин ("sex" фича) в этом сете?
data["sex"].value_counts()
```

Out[34]:
```
Male      21790
Female    10771
Name: sex, dtype: int64
```

In [35]:
```
# 2) Какой средний возраст ("age" фича) у мужчин?
data[data["sex"] == "Male"]["age"].mean()
```

Out[35]:
```
39.43354749885268
```

In [36]:
```
# 3) Какой процентаж у жителей ("native-country" фича) Италии?
print("{0:%}".format(data[data["native-country"] == "Italy"]
                      .shape[0] / data.shape[0]))
```

```
0.224195%
```

In [37]:
```
# 4-5) Каково среднее и стандартное отклонение возраста для тех, кто зарабатывает более 50 тыс. в
 год ("salary" фича)
# и тех, кто зарабатывает менее 50 тыс. в год?
ages1 = data.loc[data['salary'] == '>50K', 'age']
ages2 = data.loc[data['salary'] == '<=50K', 'age']
print("The average age of the:\nrich: {0} +- {1} years,\npoor: {2} +- {3} years.".format(
    round(ages1.mean()), round(ages1.std(), 1),
    round(ages2.mean()), round(ages2.std(), 1)))
```

```
The average age of the:
rich: 44 +- 10.5 years,
poor: 37 +- 14.0 years.
```

In [38]:
```
# 6) Правда, что все с зарплатой >50K имеют высшее образование?
if len(data.loc[data['salary'] == '>50K', 'education'].unique()) > 0:
    print("Ложь")
else:
    print("Правда")
```

```
Ложь
```

In [39]:
```
# 7.1) Отображение статистики возраста для каждой расы ("race" фича) и каждого пола ("sex" фича).
# Используйте groupby() и describe().
data.groupby(["race", "sex"])["age"].describe()
```

Out[39]:

		count	mean	std	min	25%	50%	75%	max
race	**sex**								
Amer-Indian-Eskimo	**Female**	119.0	37.117647	13.114991	17.0	27.0	36.0	46.00	80.0
	Male	192.0	37.208333	12.049563	17.0	28.0	35.0	45.00	82.0
Asian-Pac-Islander	**Female**	346.0	35.089595	12.300845	17.0	25.0	33.0	43.75	75.0
	Male	693.0	39.073593	12.883944	18.0	29.0	37.0	46.00	90.0
Black	**Female**	1555.0	37.854019	12.637197	17.0	28.0	37.0	46.00	90.0
	Male	1569.0	37.682600	12.882612	17.0	27.0	36.0	46.00	90.0
Other	**Female**	109.0	31.678899	11.631599	17.0	23.0	29.0	39.00	74.0
	Male	162.0	34.654321	11.355531	17.0	26.0	32.0	42.00	77.0
White	**Female**	8642.0	36.811618	14.329093	17.0	25.0	35.0	46.00	90.0
	Male	19174.0	39.652498	13.436029	17.0	29.0	38.0	49.00	90.0

In [40]:
```
# 7.2) Найти максимальный возраст мужчин американо-индийско-эскимосской расы (Amer-Indian-Eskimo race).
data[(data["race"] == "Amer-Indian-Eskimo") & (data["sex"] == "Male")]["age"].max()
```

Out[40]:
```
82
```

In [41]:
```
# 8) Среди кого больше доля тех, кто много зарабатывает (>50 тыс.):
# женатые или одинокие мужчины (особенность семейного положения)?
# Считать женатыми тех, кто имеет семейное положение, начиная с Married
# (Married-civ-spouse, Married-spouse-absent или Married-AF-spouse), остальные считаются холостяками.
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].isin(['Never-married',
                                   'Separated',
                                   'Divorced',
                                   'Widowed'])), 'salary'].value_counts()
```

Out[41]:
```
<=50K    7552
>50K      697
Name: salary, dtype: int64
```

In [42]:
```
data.loc[(data['sex'] == 'Male') & (data['marital-status'].str.startswith('Married')), 'salary'].value_counts()
```

Out[42]:
```
<=50K    7576
>50K     5965
Name: salary, dtype: int64
```

In [43]:

```
# 9) Какое максимальное количество часов человек работает в неделю ("hours-per-week" фича)?
# Сколько людей работает такое количество часов и каков процент тех, кто зарабатывает много среди
 них?
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 (>50K) among them {0}%".format(int(100 * rich_share)))
```

```
Max time - 99 hours./week.
Total number of such hard workers 85
Percentage of rich (>50K) among them 29%
```

In [44]:
```
# 10.1) Подсчитайте среднее время работы (hours-per-week) для тех,
# кто зарабатывает мало и много (salary) для каждой страны (native-country).
p = pd.crosstab(data["native-country"], data["salary"],
                values=data['hours-per-week'], aggfunc="mean")
p
```

Out[44]:

salary	<=50K	>50K
native-country		
?	40.164760	45.547945
Cambodia	41.416667	40.000000
Canada	37.914634	45.641026
China	37.381818	38.900000
Columbia	38.684211	50.000000
Cuba	37.985714	42.440000
Dominican-Republic	42.338235	47.000000
Ecuador	38.041667	48.750000
El-Salvador	36.030928	45.000000
England	40.483333	44.533333
France	41.058824	50.750000
Germany	39.139785	44.977273

salary	<=50K	>50K
native-country		
Greece	41.809524	50.625000
Guatemala	39.360656	36.666667
Haiti	36.325000	42.750000
Holand-Netherlands	40.000000	NaN
Honduras	34.333333	60.000000
Hong	39.142857	45.000000
Hungary	31.300000	50.000000
India	38.233333	46.475000
Iran	41.440000	47.500000
Ireland	40.947368	48.000000
Italy	39.625000	45.400000
Jamaica	38.239437	41.100000
Japan	41.000000	47.958333
Laos	40.375000	40.000000
Mexico	40.003279	46.575758
Nicaragua	36.093750	37.500000

salary	<=50K	>50K
native-country		
Outlying-US(Guam-USVI-etc)	41.857143	NaN
Peru	35.068966	40.000000
Philippines	38.065693	43.032787
Poland	38.166667	39.000000
Portugal	41.939394	41.500000
Puerto-Rico	38.470588	39.416667
Scotland	39.444444	46.666667
South	40.156250	51.437500
Taiwan	33.774194	46.800000
Thailand	42.866667	58.333333
Trinadad&Tobago	37.058824	40.000000
United-States	38.799127	45.505369
Vietnam	37.193548	39.200000
Yugoslavia	41.600000	49.500000

In [51]:
```
# 10.2) Что это будет для Италии?
p.loc["Italy"]
```

Out[51]:
```
salary
<=50K    39.625
>50K     45.400
Name: Italy, dtype: float64
```

In [52]:
```
#!pip install pandasql
```

In [53]:
```
#!python -m pip install --upgrade pip
```

In [54]:
```
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

In [55]:
```
store = pd.read_csv('./data/googleplaystore.csv')
store.columns
```

Out[55]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

In [56]:
```
sur = pd.read_csv('./data/googleplaystore_user_reviews.csv')
sur.columns
```

Out[56]:
```
Index(['App', 'Translated_Review', 'Sentiment', 'Sentiment_Polarity',
       'Sentiment_Subjectivity'],
      dtype='object')
```

Импортировали два датасета из которых будем брать информацию для запроса.

In [57]:
```
sur.head()
```

Out[57]:

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking ...	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
2	10 Best Foods for You	NaN	NaN	NaN	NaN
3	10 Best Foods for You	Works great especially going grocery store	Positive	0.40	0.875000
4	10 Best Foods for You	Best idea us	Positive	1.00	0.300000

In [58]:
```
store = store.dropna()
store = store.drop(['Reviews', 'Size', 'Installs', 'Type', 'Content Rating', 'Genres', 'Last Updated',
                    'Current Ver', 'Android Ver'], 1)
store.head()
```

Out[58]:

	App	Category	Rating	Price
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	0
1	Coloring book moana	ART_AND_DESIGN	3.9	0
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	0
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	0
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	0

In [59]:
```
sur = sur.dropna()
sur = sur.drop("Translated_Review", 1)
sur.head()
```

Out[59]:

	App	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	Positive	1.00	0.533333
1	10 Best Foods for You	Positive	0.25	0.288462
3	10 Best Foods for You	Positive	0.40	0.875000
4	10 Best Foods for You	Positive	1.00	0.300000
5	10 Best Foods for You	Positive	1.00	0.300000

In [60]:
```
store.merge(sur[["App", "Sentiment_Polarity"]], on="App").head()
```

Out[60]:

	App	Category	Rating	Price	Sentiment_Polarity
0	Coloring book moana	ART_AND_DESIGN	3.9	0	-0.250
1	Coloring book moana	ART_AND_DESIGN	3.9	0	-0.725
2	Coloring book moana	ART_AND_DESIGN	3.9	0	0.000
3	Coloring book moana	ART_AND_DESIGN	3.9	0	0.500
4	Coloring book moana	ART_AND_DESIGN	3.9	0	-0.800

In [61]:
```
%%timeit
store.merge(sur[["App", "Sentiment_Polarity"]], on="App")
```

```
20.1 ms ± 1.52 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [62]:
```
#pysqldf("""SELECT st.App, st.Category, st.Rating, sur.Sentiment_Polarity
#           FROM store AS st JOIN sur
#           ON st.App = sur.App""").head()
```

In [63]:
```
sur.groupby("App")["Sentiment_Polarity"].mean().head()
```

Out[63]:
```
App
10 Best Foods for You               0.470733
104 找工作 - 找工作 找打工 找兼職 履歷健檢 履歷診療室    0.392405
11st                                0.185943
1800 Contacts - Lens Store          0.318145
1LINE – One Line with One Touch     0.196290
Name: Sentiment_Polarity, dtype: float64
```

In [64]:
```
%%timeit
sur.groupby("App")["Sentiment_Polarity"].mean()
```

```
3.63 ms ± 129 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [65]:
```
%%timeit
pysqldf("""SELECT App, AVG(Sentiment_Polarity)
           FROM sur
           GROUP BY App
        """)
```

```
244 ms ± 80.7 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [66]:
```
pysqldf("""SELECT * FROM sur""").head()
```

Out[66]:

	App	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	Positive	1.00	0.533333
1	10 Best Foods for You	Positive	0.25	0.288462
2	10 Best Foods for You	Positive	0.40	0.875000
3	10 Best Foods for You	Positive	1.00	0.300000
4	10 Best Foods for You	Positive	1.00	0.300000