# parsing-hh-notebook

June 4, 2020

[21]:
```python
import requests
import pandas as pd
import json
import math
pd.set_option('display.max_columns', 100)
```

[44]:
```python
r = requests.get('https://api.hh.ru/vacancies', params={'text':'Data Engineer',␣
 ↪'area':'1', 'per_page':100})
per_page = r.json()['per_page']
pages = r.json()['pages']
```

[45]:
```python
vacs = []
empls = set()
for p in range(pages):
    r = requests.get('https://api.hh.ru/vacancies', params={'page': p,␣
 ↪'per_page':per_page, 'text':'Data Engineer', 'area':'1'}).json()['items']
    for i in range(len(r)):
        empls.add(r[i]['employer']['url'])
        vacs.append(r[i]['url'])
```

[46]:
```python
det_empl = []
for empl in empls:
    det_empl.append(requests.get(empl).json())
    #if len(det_empl) == 10:
    #    break

empl_df = pd.DataFrame(det_empl)[['id', 'name', 'industries']]
```

[47]:
```python
industries = []
for i in range(len(empl_df.id)):
    for j in empl_df.industries[i]:
        industries.append([empl_df.id[i], j['id'], j['name']])

ind_df = pd.DataFrame(industries, columns=['empl_id', 'id', 'name'])
empl_df.drop('industries', axis=1, inplace=True)
```

[6]:
```
det_vac = []
for vac in vacs:
    det_vac.append(requests.get(vac).json())
    if len(det_vac) % 50 == 0:
        print(f'{len(det_vac)} of {len(vacs)}')

vac_df = pd.DataFrame(det_vac)[['id', 'name', 'alternate_url', 'description',␣
 ↪'employer', 'employment', 'experience', 'key_skills', 'salary', 'schedule',␣
 ↪'specializations']]
```

```
50 of 447
100 of 447
150 of 447
200 of 447
250 of 447
300 of 447
350 of 447
400 of 447
```

[7]:
```
skills = []
for i in range(len(vac_df.id)):
    for j in vac_df.key_skills[i]:
        skills.append([vac_df.id[i], j['name']])

skill_df = pd.DataFrame(skills, columns=['vac_id', 'skill_name'])
vac_df.drop('key_skills', axis=1, inplace=True)
```

[8]:
```
vac_df = vac_df.join(pd.read_json(vac_df.salary.to_json()).T)
vac_df.drop('salary', axis=1, inplace=True)
vac_df = vac_df.join(pd.read_json(vac_df.employer.to_json()).T['id'],␣
 ↪rsuffix='_empl')
vac_df.drop('employer', axis=1, inplace=True)
vac_df.rename(columns={'id_empl':'empl_id'}, inplace=True)
```

[9]:
```
def calc_salary(from_, to_, gross, curr):
  if from_ == None:
    from_ = float('NaN')
  if to_ == None:
    to_ = float('NaN')
  if math.isnan(from_) and math.isnan(to_) or gross == None:
    res = float('NaN')
  if math.isnan(from_):
    if gross == False:
      res = to_ / 0.87
    else:
      res = to_
  elif math.isnan(to_):
    if gross == False:
      res = from_ / 0.87
    else:
      res = from_
  else:
    res = (from_ + to_) / 2
    if gross == False:
      res /= 0.87
  if curr == 'USD' :
    res *= 64.3
  elif curr == 'EUR':
    res *= 70.85

  return res

vac_df['salary'] = vac_df.apply(lambda x: calc_salary(x['from'], x['to'],␣
 ↪x['gross'], x['currency']), axis=1)
vac_df.drop(['from', 'to', 'gross', 'currency'], axis=1, inplace=True)
```

[10]:
```
empl_df.to_csv('empl.csv', header=True, index=False)
ind_df.to_csv('ind.csv', header=True, index=False)
skill_df.to_csv('skills.csv', header=True, index=False)
vac_df[['id', 'name', 'salary', 'empl_id']].to_csv('vac.csv', header=True,␣
 ↪index=False)
```

1.      DE:                 ,                ,

[48]:
```
print("            : " +   str(len(vac_df)))
print("           : " + str(len(ind_df.groupby(['name']))))
print("         : " + str(len(empl_df)))
```

```
          : 447
         : 116
       : 218
```

[17]:
```
print(empl_df)
```

```
          id                             name
0       2180                             Ozon
1       3529
2      11680                                .
3       1846                  Brainpower CIS
4       1304                          Luxoft
..       …                               …
213     3095
214     1776              McKinsey & Company
215  2651877                             CTI
216  1038826  Business and Technology Services
```

```
217     4496

[218 rows x 2 columns]
```

2.      :          ,       ,

[78]:
```python
skill_df.groupby('skill_name').size().reset_index(name='counts').
↪sort_values(['counts'], ascending=False).head(20)
```

[78]:
```
             skill_name  counts
357              Python     132
393                 SQL     110
230                Java      66
254               Linux      59
563                       57
178                 Git      43
343          PostgreSQL      32
196              Hadoop      28
120         Data Mining      25
45       Atlassian Jira      25
420               Spark      24
68                  C++      24
272              MS SQL      24
57             Big Data      23
387               SCALA      21
234          JavaScript      19
143              Docker      16
352  Project management      16
319              ORACLE      14
297             MongoDB      14
```

[81]:
```python
print("             :")
print("Python: " + str(len(skill_df.loc[skill_df['skill_name'] == "Python"])))
print("Java: " + str(len(skill_df.loc[skill_df['skill_name'] == "Java"])))
print("Kotlin: " + str(len(skill_df.loc[skill_df['skill_name'] == "Kotlin"])))
print("C++: " + str(len(skill_df.loc[skill_df['skill_name'] == "C++"])))
print("           :")
print("Linux: " + str(len(skill_df.loc[skill_df['skill_name'] == "Linux"])))
print("     :")
print("Docker: " + str(len(skill_df.loc[skill_df['skill_name'] == "Docker"])))
print("SQL: " + str(len(skill_df.loc[skill_df['skill_name'] == "SQL"])))
print("Git: " + str(len(skill_df.loc[skill_df['skill_name'] == "Git"])))
```

```
             :
Python: 132
Java: 66
Kotlin: 2
C++: 24
```

```
          :
Linux: 59
     :
Docker: 16
SQL: 110
Git: 43
```

3. ,

,

.

```
[ ]:
```