Введение

Добро пожаловать! Сегодня мы разберем некоторые вопросы и задания, которые позволяют определить на собеседовании, знаете ли вы на начальном уровне язык программирования Python. Вы можете самостоятельно пройти все эти задания. Для этого вам необходимо скачать архив из репозитория USA университеты

Вам нужен файл cc_institution_grads. Список заданий представлен ниже:

- Напечатайте данный лист в обратном порядке
  `my_list=[1,2,3,4,5]`
- Считайте данные из файла cc_institution_grads.csv в переменную df
- Какие гендеры есть в данной таблице? Выведите все
- Выведите топ 5 лет, которые встречаются в БД (Используя groupby, и как сделать без него)
- Выведите топ 5 рас, встречающихся среди Женщин.
- Есть база данных. используя ее, найдите среднее значение возраста, и максимальное.
  `db2 = sns.load_dataset('titanic')`
- Укажите процент выживших в аварии
- Создайте таблицу, в которой будет указано: минимальный возраст каждого пола, максимальный возраст каждого пола
- Найдите, в каком классе мест пассажиров разница возрастов между самым старшим и самым младшим максимальна.
- Есть два датафрейма. Объедините их с помощью Left join метода (то есть чтобы все значения из левой таблицы остались, даже если у правой таблицы нет значений для каких-то наименований)

```
df1 = pd.DataFrame({'lkey': ['rabbit', 'wolf', 'parrot', 'rabbit','rabbit', 'wolf',
'parrot', 'rabbit','elephant'],
                    'kg': [1.2, 12.6, 0.4, 1.8,1.9, 11.1, 0.2, 1.6,340.4]})
df2 = pd.DataFrame({'rkey': ['rabbit', 'wolf', 'parrot','dolphin'],
                    'diet': ['carrot', 'meat', 'seeds','fish']})
```

Пройдя эти задания, проверяющему вы покажете, что вы:

- [x] ~~Умеете работать с БД~~
- [x] ~~Умеете использовать `.groupby` `.agg` `.merge` `.unique` `Counter` `reversed`~~

> Я обнаружил для себя, что еще плохо знаю язык Markdown, поэтому вот статья по написанию отличных файлов README

```
# Напечатайте лист в обратном порядке
```

```
my_list=[1,2,3,4,5]
```

```
#Метод 1
print(my_list[::-1])
#Метод 2
q=list(reversed(my_list))
print(q)
```

```
[5, 4, 3, 2, 1]
[5, 4, 3, 2, 1]
```

```
#считайте данные из файла cc_institution_grads.csv в переменную df
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("cc_institution_grads.csv", sep=',')
df
```

	index	unitid	year	gender	race	cohort	grad_cohort	grad_100	grad_150	grad_
0	0	100760	2011	B	X	2y all	446.0	73.0	105.0	16.4
1	1	100760	2011	M	X	2y all	185.0	NaN	40.0	NaN
2	2	100760	2011	F	X	2y all	261.0	NaN	65.0	NaN
3	3	100760	2011	B	W	2y all	348.0	NaN	86.0	NaN
4	4	100760	2011	M	W	2y all	162.0	NaN	35.0	NaN
...	...	...	...	...	...	...	...	...	...	...
1302097	1302097	168591	2002	F	Ai	4y other	NaN	NaN	NaN	NaN
1302098	1302098	168740	2002	F	Ai	4y other	0.0	NaN	0.0	NaN
1302099	1302099	169716	2002	F	Ai	4y other	NaN	NaN	NaN	NaN
1302100	1302100	170082	2002	F	Ai	4y other	NaN	NaN	NaN	NaN
1302101	1302101	170091	2002	F	Ai	4y other	0.0	NaN	0.0	NaN

1302102 rows × 11 columns

```
#Какие гендеры есть в данной таблице
```

```
#Метод 1
print(list(set(df.gender)))
#Метод 2
print(df.gender.unique())
```

```
['B', 'F', 'M']
['B' 'M' 'F']
```

```
#выведите топ 5 лет, которые встречаются в БД
```

```
#Метод 1
from collections import Counter
```

```
count5=Counter(df['year'])
count5=count5.most_common(5)
for i in range (5):
    print(count5[i][0])
print('---------------')
#Метод 2
df.groupby('year', as_index=False) \
    .agg({'index':'count'}) \
    .sort_values(by='year', ascending=False) \
    .head(5)
```

```
2011
2012
2013
2010
2009
---------------
```

	year	index
11	2013	110466
10	2012	110466
9	2011	110466
8	2010	107856
7	2009	107856

```
#выведите топ 5 рас, встречающихся среди Женщин.
```

```
df[df.gender=='F'].groupby('race', as_index=False) \
    .agg({'gender':'count'}) \
    .sort_values(by='gender', ascending=False) \
    .head(5)
```

	race	gender
0	A	72339
1	Ai	72339
2	B	72339
3	H	72339
4	W	72339

```
#Есть база данных. используя ее, найдите среднее значение возраста, и максимальное.
db2 = sns.load_dataset('titanic')
db2.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	

```
db2.age.agg(['max','mean'])
```

```
max     80.000000
mean    29.699118
Name: age, dtype: float64
```

```
# Укажите процент выживших в аварии
```

```
str(int((db2.survived[db2.survived==1].agg('count')/db2.survived.agg('count'))*100))+'%'
```

```
'38%'
```

```
#Создайте таблицу, в которой будет указано: минимальный возраст каждого пола, максимальный возраст каждого пола
```

```
db2.groupby('sex', as_index=False).agg({'age':['min','max']})
```

	sex	age	
		min	max
0	female	0.75	63.0
1	male	0.42	80.0

```
#Найдите, в каком классе мест пассажиров разница возрастов между самым старшим и самым младшим максимальна.
```

```
db2.groupby('class', as_index=False).age.max().age-db2.groupby('class', as_index=False).age.min().age
```

```
0    79.08
1    69.33
2    73.58
Name: age, dtype: float64
```

```
delta=db2.groupby('class', as_index=False).age.max() #Делаем ДФ с максимальными возрастами для каждого
delta=delta.rename({'age':'age_old'}, axis=1) #Переименуем для удобства
delta['age_young']=db2.groupby('class', as_index=False).age.min().age #Добавляем колонку с минимальными возрастами
delta['age_d']=delta['age_old']-delta['age_young'] # добавляем третью с вычетом
print(delta)
print(f"Answer is {delta[delta.age_d==delta.age_d.max()]['class'][0]} class") #Выводим то значение, у которого разность возрастов будет совпадать с максимальным значением из списка
```

```
    class  age_old  age_young  age_d
0   First     80.0       0.92  79.08
1  Second     70.0       0.67  69.33
2   Third     74.0       0.42  73.58
Answer is First class
```

```
# Есть два датафрейма. Объедините их с помощью Left join метода (то есть чтобы все значения из левой таблицы остались, даже если у правой таблицы нет значений для каких-то наименований)
```

```
df1 = pd.DataFrame({'lkey': ['rabbit', 'wolf', 'parrot', 'rabbit','rabbit', 'wolf', 'parrot', 'rabbit','elephant'],
                    'kg': [1.2, 12.6, 0.4, 1.8,1.9, 11.1, 0.2, 1.6,340.4]})
df2 = pd.DataFrame({'rkey': ['rabbit', 'wolf', 'parrot','dolphin'],
                    'diet': ['carrot', 'meat', 'seeds','fish']})
```

```
df1
```

	lkey	kg
0	rabbit	1.2
1	wolf	12.6
2	parrot	0.4
3	rabbit	1.8
4	rabbit	1.9
5	wolf	11.1
6	parrot	0.2
7	rabbit	1.6
8	elephant	340.4

```
df2
```

	rkey	diet
0	rabbit	carrot
1	wolf	meat
2	parrot	seeds
3	dolphin	fish

```
#Если просто написать merge, то у нас пропадут некоторые данные, так как у нас было животное baz в первой таблице
df1.merge(df2, left_on='lkey', right_on='rkey')
```

	lkey	kg	rkey	diet
0	rabbit	1.2	rabbit	carrot
1	rabbit	1.8	rabbit	carrot
2	rabbit	1.9	rabbit	carrot
3	rabbit	1.6	rabbit	carrot

	lkey	kg	rkey	diet
4	wolf	12.6	wolf	meat
5	wolf	11.1	wolf	meat
6	parrot	0.4	parrot	seeds
7	parrot	0.2	parrot	seeds

```
result=df1.merge(df2, left_on='lkey', right_on='rkey', how='left').drop('rkey',
axis=1).rename({'lkey':'animal'},axis=1)
result
```

	animal	kg	diet
0	rabbit	1.2	carrot
1	wolf	12.6	meat
2	parrot	0.4	seeds
3	rabbit	1.8	carrot
4	rabbit	1.9	carrot
5	wolf	11.1	meat
6	parrot	0.2	seeds
7	rabbit	1.6	carrot
8	elephant	340.4	NaN