Анализ данных с Python

Лабораторная работа 2 – Pandas DataFrame: группировка данных и сводные таблицы

Создадим новый Jupiter ноутбук lab_2.ipynb

Продублируем первую ячейку из предыдущей работы, в которой создается датафрейм:

```
import pandas as pd

team = [1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 3, 3, 2, 3, 3]

name = ['Воронюк', 'Денисенко', 'Дуплій', 'Іванов', 'Капітан',
        'Карпова', 'Кіріенко', 'Коваленко', 'Луговий', 'Петренко',
        'Петров', 'Савчук', 'Сорокіна', 'Старчук', 'Шульга']

category = [4, 5, 6, 6, 4, 5, 5, 6, 4, 4, 5, 5, 5, 6, 6]

days = [24, 26, 20, 24, 25, 25, 26, 24, 20, 27, 12, 24, 24, 23, 24]

total = [600.0, 840.0, 800.0, 960.0, 650.0, 780.0, 840.0, 960.0, 500.0,
         750.0, 360.0, 720.0, 720.0, 920.0, 960.0]

df = pd.DataFrame( data = {
                'team': team,
                'name': name,
                'category': category,
                'days': days,
                'total': total
       })

df
```

Подсчитаем сколько у нас строк по каждой бригаде, используя метод **value_counts**:

```
df['team'].value_counts()
```

```
2    6
1    5
3    4
Name: team, dtype: int64
```

Видим что 6 чел. во 2-й бригаде, 5 – во 1-й и 4 в 3-й.

Если хотим выводить по номерам бригад, то вот так:

```
df['team'].value_counts().sort_index()
```

```
1    5
2    6
3    4
Name: team, dtype: int64
```

Теперь попробуем сгруппировать сумму зарплаты по бригадам. Применим метод **groupby**

```
df.groupby('team')['total'].sum()
```

```
team
1    3680.0
2    4720.0
3    2960.0
Name: total, dtype: float64
```

Если нас интересует помимо зарплаты еще суммарное количество отработанных дней:

```
df.groupby('team')[['total', 'days']].sum()
```

team	total	days
1	3680.0	119
2	4720.0	146
3	2960.0	83

А теперь отсортируем вывод по убыванию зарплаты:

```
df.groupby('team')[['total', 'days']].sum().sort_values('total', ascending=False)
```

team	total	days
2	4720.0	146
1	3680.0	119
3	2960.0	83

(если по возрастанию, то **ascending=True**)

Кроме сумм мы может получить средние значения, используя **mean()** вместо **sum()**:

```
df.groupby('team')[['total', 'days']].mean()
```

team	total	days
1	736.000000	23.800000
2	786.666667	24.333333
3	740.000000	20.750000

Теперь перейдем к созданию сводных таблиц (**pivot_table**).

Допустим, мы хотим получить анализ зарплаты в разрезе «бригада-разряд» в табличном виде, где номера бригады будут в строках таблицы, номера разрядов – в колонках:

```
df.pivot_table(values='total', index='team', columns='category', aggfunc='sum', fill_value=0)
```

category	4	5	6
team			
1	1100	1620	960
2	1400	1560	1760
3	0	1080	1880

То же самое, но с итогами по строкам и колонкам:

```
df.pivot_table(values='total', index='team', columns='category',
               aggfunc='sum', fill_value=0, margins=True)
```

category	4	5	6	All
team				
1	1100	1620	960	3680.0
2	1400	1560	1760	4720.0
3	0	1080	1880	2960.0
All	2500	4260	4600	11360.0

Поменяем сумму на среднее значение:

```
df.pivot_table(values='total', index='team', columns='category',
               aggfunc='mean', fill_value=0, margins=True)
```

category	4	5	6	All
team				
1	550	810	960	736.000000
2	700	780	880	786.666667
3	0	540	940	740.000000
All	625	710	920	757.333333

А теперь и сумма, и среднее:

```
df.pivot_table(values='total', index='team', columns='category',
               aggfunc=['sum', 'mean'], fill_value=0, margins=True)
```

	sum				mean			
category	4	5	6	All	4	5	6	All
team								
1	1100	1620	960	3680.0	550	810	960	736.000000
2	1400	1560	1760	4720.0	700	780	880	786.666667
3	0	1080	1880	2960.0	0	540	940	740.000000
All	2500	4260	4600	11360.0	625	710	920	757.333333

Индивидуальное задание

Продолжить исследование своего датафрейма, используя рассмотренные здесь функции count_values, grpupby и pivot_table