Лабораторная работа № 5

Перед началом выполнения каждой лабораторной работы, необходимо выполнить код в двух ячейках ниже. В случае если модуль `datetime` не установлен нужно установить. В последнем принте, вывести свою фамилию и инициалы.

In [408]:

```
!whoami
```

```
rogozhina\home
```

In [409]:

```
from datetime import datetime

current_time = datetime.now()
print(current_time)
print("Rogozhina N.A.") # написать здесь свою фамилию и инициалы
```

```
2022-12-20 22:41:51.774844
Rogozhina N.A.
```

In [39]:

```
import pandas as pd
import numpy as np
import scipy.stats as sps

import warnings
warnings.simplefilter("ignore", FutureWarning)
```

Задания

Задание 1. Создайте любые три датафрейма и объедините их функциями `pd.concat` , `pd.merge` и `df.join` (т.е. 3-мя способами). Объясните есть ли разница между результатами полученными разными функциями.

In [40]:

```
df1 = pd.DataFrame(sps.uniform.rvs(size=(3, 3)),
                   columns=['one', 'two', 'three'])

df2 = pd.DataFrame(sps.uniform.rvs(size=(3, 3)),
                   columns=['one', 'two', 'three'])

df3 = pd.DataFrame(sps.uniform.rvs(size=(3, 3)),
                   columns=['one', 'two', 'three'])
print(df1)
print(df2)
print(df3)

# У нас вывелись изначальные датафреймы
```

```
        one       two     three
0  0.305297  0.400003  0.869773
1  0.714676  0.240318  0.141650
2  0.447152  0.234423  0.471918
        one       two     three
0  0.312278  0.242544  0.355362
1  0.180426  0.176411  0.195842
2  0.925838  0.653403  0.556334
        one       two     three
0  0.229169  0.424593  0.647451
1  0.235025  0.839006  0.354060
2  0.097690  0.728964  0.586160
```

In [71]:

```
a = pd.merge(df1,df2,how='outer')
b = df1.join(df2,how='outer', on='one',lsuffix='l',rsuffix='r')

print(f"""Через метод '.concat':\n
{pd.concat([df1,df2,df3],keys=['df1', 'df2', 'df3'])}\n
Через метод '.merge':\n
{pd.merge(a,df3,how='outer')}\n
Через метод '.join':\n
{b}\n
{b.join(df3,how='outer', on='one',lsuffix='?',rsuffix='!')}\n""")
```

```
Через метод '.join':

          one      onel      twol    threel      oner      twor    threer
0.0  0.305297  0.305297  0.400003  0.869773       NaN       NaN       NaN
1.0  0.714676  0.714676  0.240318  0.141650       NaN       NaN       NaN
2.0  0.447152  0.447152  0.234423  0.471918       NaN       NaN       NaN
NaN  0.000000       NaN       NaN       NaN  0.312278  0.242544  0.355362
NaN  1.000000       NaN       NaN       NaN  0.180426  0.176411  0.195842
NaN  2.000000       NaN       NaN       NaN  0.925838  0.653403  0.556334

          one      one?      onel      twol    threel      oner      twor  \
0.0  0.305297  0.305297  0.305297  0.400003  0.869773       NaN       NaN
1.0  0.714676  0.714676  0.714676  0.240318  0.141650       NaN       NaN
2.0  0.447152  0.447152  0.447152  0.234423  0.471918       NaN       NaN
NaN  0.000000  0.000000       NaN       NaN       NaN  0.312278  0.242544
NaN  1.000000  1.000000       NaN       NaN       NaN  0.180426  0.176411
NaN  2.000000  2.000000       NaN       NaN       NaN  0.925838  0.653403

       threer      one!       two     three
0.0       NaN       NaN       NaN       NaN
```

Таким образом, мы видим, как работают разные методы при объединении датафреймов.

1. Метод **.concat**, в отличии от join и merge, которые по умолчанию работают со столбцами, позволяет выбрать, выполнять объединение по столбцам или по строкам.
2. Метод **.merge** объединяет все столбцы из двух таблиц с общими столбцами, переименованными в определенные suffixes.
3. Метод **.join** объединит все столбцы из двух таблиц с общими столбцами, переименованными в определенные lsuffix и rsuffix. Способ объединения строк из двух таблиц определяется с помощью how – inner, outer, right, left. В данном примере был использован способ объединения **outer**, который вывел все что есть в наших датафреймах.

Если использовать другие способы объединения:

a. **inner** - ищет пересечения (логич. И)

b. **left join** - выведет то, что входит в left, и часть right, которая тоже подходит в left.

c. **right join** - как и в предыдущем способе, только наоборот.

Задание 2. `pd.DataFrame` , замените случайные 10% элементов на пропуски (`np.nan`), а затем добавьте по столбцу для оценок первых 4 моментов кумулятивно —

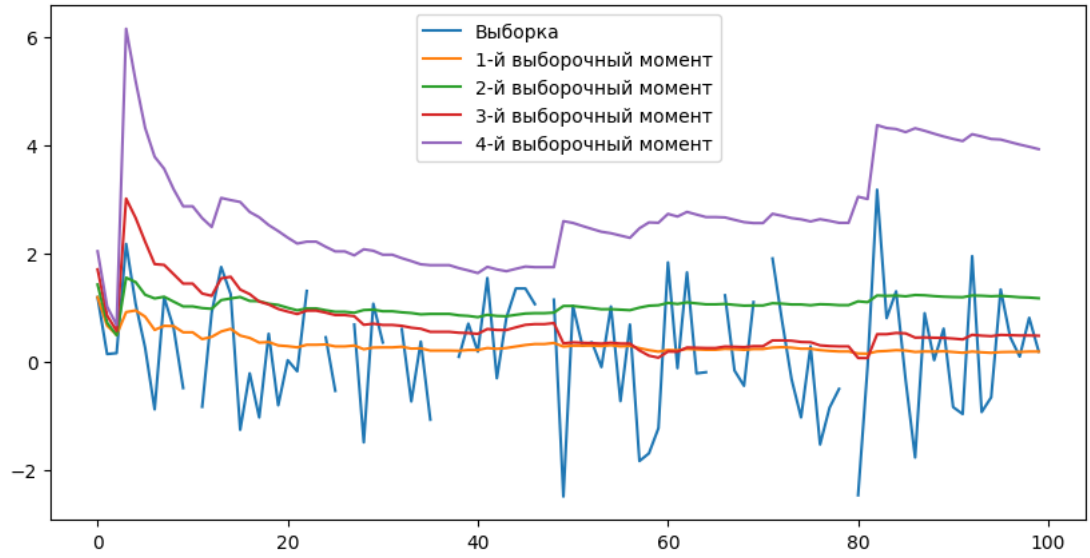
$$\frac{1}{m}\sum_{i=1}^{m} X_i^k,\ i \in \overline{1,m},\ m \in \overline{1,n},\ k \in \overline{1,4}$$

Ваша функция должна корректно обрабатывать пропуски. В конце постройте график.

Формулировка задания мне не понятна.

In [80]:

```
import numpy as np
import pandas as pd
import scipy.stats as sps
# генерируем выборку
n = 100
sample = sps.norm.rvs(size=n)

# создаем пропуски
index = np.random.choice(np.arange(n), int(0.1 * n), replace=True) # Случайная выборка из значений заданного одномерного массива.
sample[index] = np.nan

# заводим dataframe

df = pd.DataFrame(sample, columns = ['Выборка'])
# создайте датафрейм на основе данных выше и задайте название столбцу Выборка

for k in range(1,5):
    df['{}-й выборочный момент'.format(k)] = (
        df['Выборка'] ** k
    ).expanding().mean()

df.plot(figsize=(10, 5));
```



In [24]:

```
df = pd.DataFrame({"B": [2, 1, 2, 5, np.nan, 4]})
```

In [25]:

```
df
```

Out[25]:

	B
0	2.0
1	1.0
2	2.0
3	5.0
4	NaN
5	4.0

In [30]:

```
# пример как работает expanding
df.expanding().sum()
```

Out[30]:

	B
0	2.0
1	3.0
2	5.0
3	10.0
4	10.0
5	14.0

In [104]:

```
#Функция nanmean() вычисляет среднее арифметическое значений элементов массива, игнорируя значения np.nan.
df.expanding().apply(np.nanmean)
```

Out[104]:

	B
0	2.000000
1	1.500000
2	1.666667
3	2.500000
4	2.500000
5	2.800000

In [106]:

```
# тот же результат
df.expanding().mean()
```

Out[106]:

	B
0	2.000000
1	1.500000
2	1.666667
3	2.500000
4	2.500000
5	2.800000

Задание 3. Из датафрейма `df_host` ниже, используя функцию `pd.pivot_table` во всех пунктых определите:

3.1. Среднюю зарплату по специальностям.

3.2. По какой специалиности самая высокая средняя зарплата.

3.3. Определите максимальный и минимальный стаж по каждой специальности.

3.4.(доп)* Определите по каждой специальности самого высокооплачиваемого специалиста.

In [401]:

```
df_host = pd.DataFrame({
    'Специальность' : ['Менеджер', 'Врач',
                       'Учитель', 'Психолог', 'Повар'] * 6,
    'Специалист' : ['Александр', 'Иван', 'Светлана'] * 10,
    'Стаж' : sps.randint(low=2, high=25).rvs(size=30),
    'Зарплата': sps.randint(low=30000, high=300000).rvs(size=30)
})
```

In [402]:

```
df_host
```

2	Учитель	Светлана	10	241925
3	Психолог	Александр	4	203691
4	Повар	Иван	11	135863
5	Менеджер	Светлана	7	175837
6	Врач	Александр	16	187566
7	Учитель	Иван	14	244274
8	Психолог	Светлана	16	164518
9	Повар	Александр	13	200087
10	Менеджер	Иван	23	273802
11	Врач	Светлана	8	297544
12	Учитель	Александр	4	80615
13	Психолог	Иван	7	67646
14	Повар	Светлана	3	123336

In [403]:

```
#3.1 Средняя зарплата по специальностям

pd.pivot_table(df_host, columns = 'Специальность',values = 'Зарплата', aggfunc = np.mean)
```

Out[403]:

Специальность	Врач	Менеджер	Повар	Психолог	Учитель
Зарплата	182669.0	183872.666667	164082.5	124000.0	163164.833333

In [404]:

```
#3.2 По какой специальности самая высокая средняя зарплата?

a = pd.pivot_table(df_host, columns = 'Специальность',values = 'Зарплата', aggfunc = np.mean)
for i in a.values:
    arr = list(i)

b = max(arr)

for i in a.columns:
    if b == a.at['Зарплата',i]:
        print(f"""Специальность: {i}\nСредняя зарплата: {b}""")
```

```
Специальность: Менеджер
Средняя зарплата: 183872.66666666666
```

In [405]:

```
#3.3 Максимальный и минимальный стаж по каждой специальности:

# Создадим список со специальностями:
work = list(set(df_host['Специальность']))

# Определим две таблицы с максимальным и минимальным стажем:
max_st = (pd.pivot_table(df_host, columns = 'Специальность',values = 'Стаж', aggfunc = np.max)).T
min_st = (pd.pivot_table(df_host, columns = 'Специальность',values = 'Стаж', aggfunc = np.min)).T

max_st = max_st.to_dict('index')
min_st = min_st.to_dict('index')

# Создадим пустую таблицу:
res = pd.DataFrame({
    "Max":0,
    "Min":0}, index = work)

# Добавим значения
for i in work:
    res.at[i,'Max'] = max_st[i]['Стаж']
    res.at[i,'Min'] = min_st[i]['Стаж']

res
```

Out[405]:

	Max	Min
Учитель	24	3
Менеджер	23	5
Психолог	16	4
Повар	21	3
Врач	24	5

In [406]:

```
#3.4.(доп)* Определите по каждой специальности самого высокооплачиваемого специалиста.

# Для начала выведем табличку, в которой будут только самые высокие зарплаты у каждого работника каждой специальности.
# Для удобства я транспонировала данные в таблице

(pd.pivot_table(df_host,index = 'Специалист', columns = 'Специальность', values = 'Зарплата', aggfunc = np.max)).T
```

Out[406]:

Специалист	Александр	Иван	Светлана
Специальность			
Врач	187566	246668	297544
Менеджер	141090	273802	250014
Повар	200087	138538	228268
Психолог	203691	67646	164518
Учитель	179804	244274	241925

In [407]:

```
# Найдем самых высокооплачиваемых специалистов.
# Для этого в библиотеке Pandas есть функция dataframe.idxmax()

a = (pd.pivot_table(df_host,index = 'Специалист', columns = 'Специальность', values = 'Зарплата', aggfunc = np.max)).T

MaxValueIndex = a.idxmax(axis=1)

MaxValueIndex
```

Out[407]:

```
Специальность
Врач          Светлана
Менеджер          Иван
Повар         Светлана
Психолог     Александр
Учитель           Иван
dtype: object
```

In [ ]:

In [ ]:

```
```