Выполнил Родионов Д.А. ИУ5-65 2 задание 4 сет

In [4]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
states = '/Users\Dmitry\Downloads/states_all.csv'
data = pd.read_csv(states, sep=",")
```

In [5]:
```python
data
```

Out[5]:

	PRIMARY_KEY	STATE	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_I
0	1992_ALABAMA	ALABAMA	1992	NaN	2678885.0	304177.0	1659028.0	715680.0	
1	1992_ALASKA	ALASKA	1992	NaN	1049591.0	106780.0	720711.0	222100.0	
2	1992_ARIZONA	ARIZONA	1992	NaN	3258079.0	297888.0	1369815.0	1590376.0	
3	1992_ARKANSAS	ARKANSAS	1992	NaN	1711959.0	178571.0	958785.0	574603.0	
4	1992_CALIFORNIA	CALIFORNIA	1992	NaN	26260025.0	2072470.0	16546514.0	7641041.0	
...	...	...	...	...	...	...	...	...	
1710	2019_VIRGINIA	VIRGINIA	2019	NaN	NaN	NaN	NaN	NaN	
1711	2019_WASHINGTON	WASHINGTON	2019	NaN	NaN	NaN	NaN	NaN	
1712	2019_WEST_VIRGINIA	WEST_VIRGINIA	2019	NaN	NaN	NaN	NaN	NaN	
1713	2019_WISCONSIN	WISCONSIN	2019	NaN	NaN	NaN	NaN	NaN	
1714	2019_WYOMING	WYOMING	2019	NaN	NaN	NaN	NaN	NaN	

1715 rows × 25 columns

In [6]:
```python
data.describe()
```

Out[6]:

	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENDITURE	INSTRUCTI
count	1715.000000	1.224000e+03	1.275000e+03	1.275000e+03	1.275000e+03	1.275000e+03	1.275000e+03	
mean	2002.075219	9.175416e+05	9.102045e+06	7.677799e+05	4.223743e+06	4.110522e+06	9.206242e+06	
std	9.568621	1.066514e+06	1.175962e+07	1.146992e+06	5.549735e+06	5.489562e+06	1.199279e+07	
min	1986.000000	4.386600e+04	4.656500e+05	3.102000e+04	0.000000e+00	2.209300e+04	4.816650e+05	
25%	1994.000000	2.645145e+05	2.189504e+06	1.899575e+05	1.165776e+06	7.151210e+05	2.170404e+06	
50%	2002.000000	6.499335e+05	5.085826e+06	4.035480e+05	2.537754e+06	2.058996e+06	5.242672e+06	
75%	2010.000000	1.010532e+06	1.084516e+07	8.279320e+05	5.055548e+06	4.755293e+06	1.074420e+07	
max	2019.000000	6.307022e+06	8.921726e+07	9.990221e+06	5.090457e+07	3.610526e+07	8.532013e+07	

8 rows × 23 columns

In [7]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1715 entries, 0 to 1714
Data columns (total 25 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   PRIMARY_KEY                   1715 non-null   object 
 1   STATE                         1715 non-null   object 
 2   YEAR                          1715 non-null   int64  
 3   ENROLL                        1224 non-null   float64
 4   TOTAL_REVENUE                 1275 non-null   float64
 5   FEDERAL_REVENUE               1275 non-null   float64
 6   STATE_REVENUE                 1275 non-null   float64
 7   LOCAL_REVENUE                 1275 non-null   float64
 8   TOTAL_EXPENDITURE             1275 non-null   float64
 9   INSTRUCTION_EXPENDITURE       1275 non-null   float64
 10  SUPPORT_SERVICES_EXPENDITURE  1275 non-null   float64
 11  OTHER_EXPENDITURE             1224 non-null   float64
 12  CAPITAL_OUTLAY_EXPENDITURE    1275 non-null   float64
 13  GRADES_PK_G                   1542 non-null   float64
 14  GRADES_KG_G                   1632 non-null   float64
 15  GRADES_4_G                    1632 non-null   float64
 16  GRADES_8_G                    1632 non-null   float64
 17  GRADES_12_G                   1632 non-null   float64
 18  GRADES_1_8_G                  1020 non-null   float64
 19  GRADES_9_12_G                 1071 non-null   float64
 20  GRADES_ALL_G                  1632 non-null   float64
 21  AVG_MATH_4_SCORE              565 non-null    float64
 22  AVG_MATH_8_SCORE              602 non-null    float64
 23  AVG_READING_4_SCORE           650 non-null    float64
 24  AVG_READING_8_SCORE           562 non-null    float64
dtypes: float64(22), int64(1), object(2)
memory usage: 335.1+ KB
```

Пропусков в категориальных данных обнаружено не было. Заполним пропуски у поля GRADES_KG_G, так как в этом столбце их не так много.

In [6]:
```
d_gr = data[['GRADES_KG_G']]
d_gr
```

Out[6]:

	GRADES_KG_G
0	55460.0
1	10152.0
2	53497.0
3	33511.0
4	431763.0
...	...
1710	NaN
1711	NaN
1712	NaN
1713	NaN
1714	NaN

1715 rows × 1 columns

In [7]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [8]:
```
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(d_gr)
mask_missing_values_only
```

Out[8]:
```
array([[False],
       [False],
       [False],
       ...,
       [ True],
       [ True],
       [ True]])
```

In [9]:
```
strategies=['mean', 'median', 'most_frequent']
```

При сравнении трех стратегий, была выбрана стратегия 'mean'

In [10]:
```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(d_gr)
    return data_num_imp[mask_missing_values_only]
```

In [15]:
```
strategies[0], test_num_impute(strategies[0])
```

Out[15]:
```
('mean',
 array([68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098, 68810.9620098,
        68810.9620098, 68810.9620098, 68810.9620098]))
```

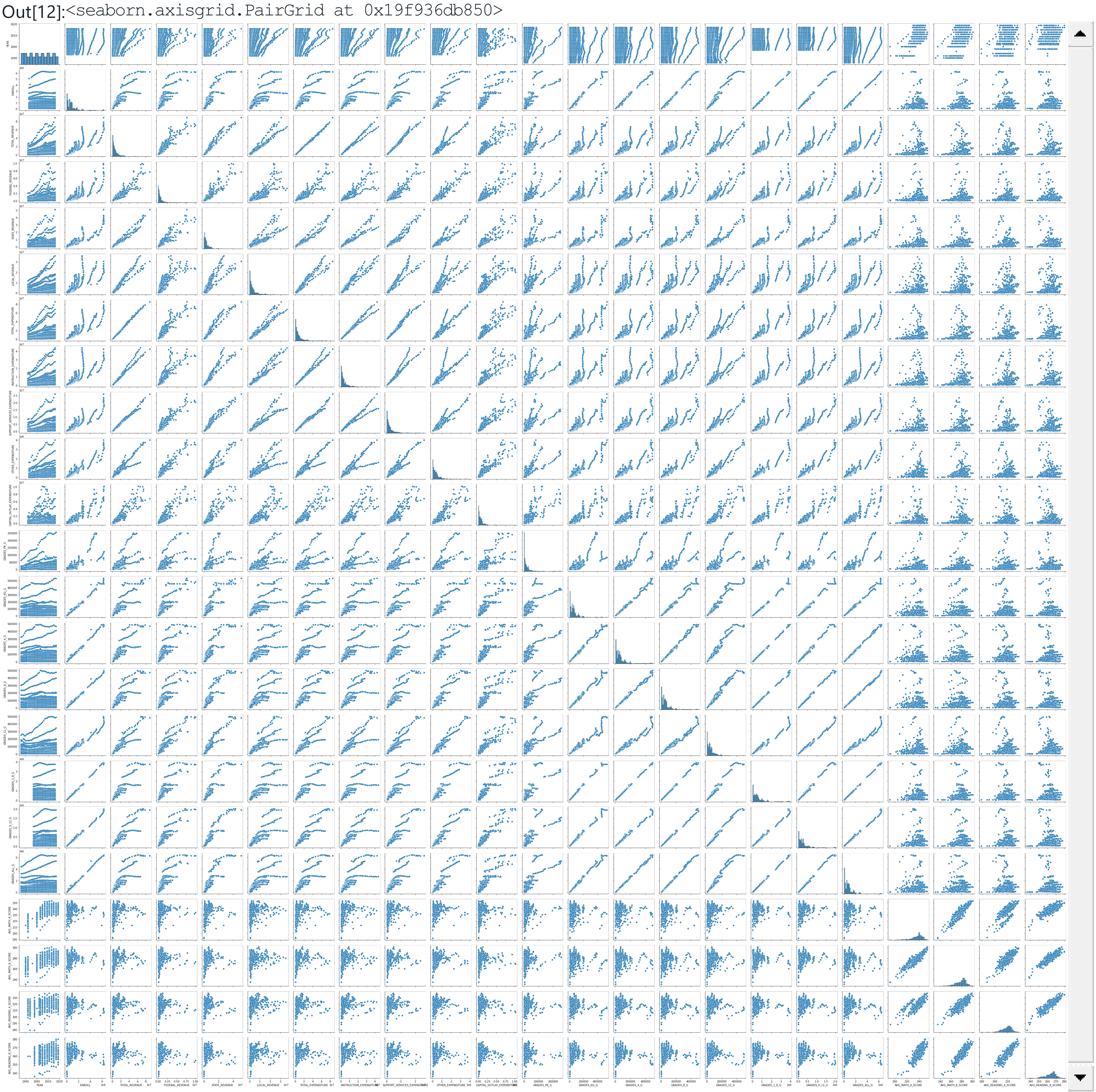
Таким образом, была использована импутация. В ходе реализации была использован метод SimpleImputor, использущий страгию "Среднее значение". При сравнении значений трех стратегий, именно "mean" подходил лучше всего.

In [11]:
```
data.columns
```

Out[11]:
```
Index(['PRIMARY_KEY', 'STATE', 'YEAR', 'ENROLL', 'TOTAL_REVENUE',
       'FEDERAL_REVENUE', 'STATE_REVENUE', 'LOCAL_REVENUE',
       'TOTAL_EXPENDITURE', 'INSTRUCTION_EXPENDITURE',
       'SUPPORT_SERVICES_EXPENDITURE', 'OTHER_EXPENDITURE',
       'CAPITAL_OUTLAY_EXPENDITURE', 'GRADES_PK_G', 'GRADES_KG_G',
       'GRADES_4_G', 'GRADES_8_G', 'GRADES_12_G', 'GRADES_1_8_G',
       'GRADES_9_12_G', 'GRADES_ALL_G', 'AVG_MATH_4_SCORE', 'AVG_MATH_8_SCORE',
       'AVG_READING_4_SCORE', 'AVG_READING_8_SCORE'],
      dtype='object')
```

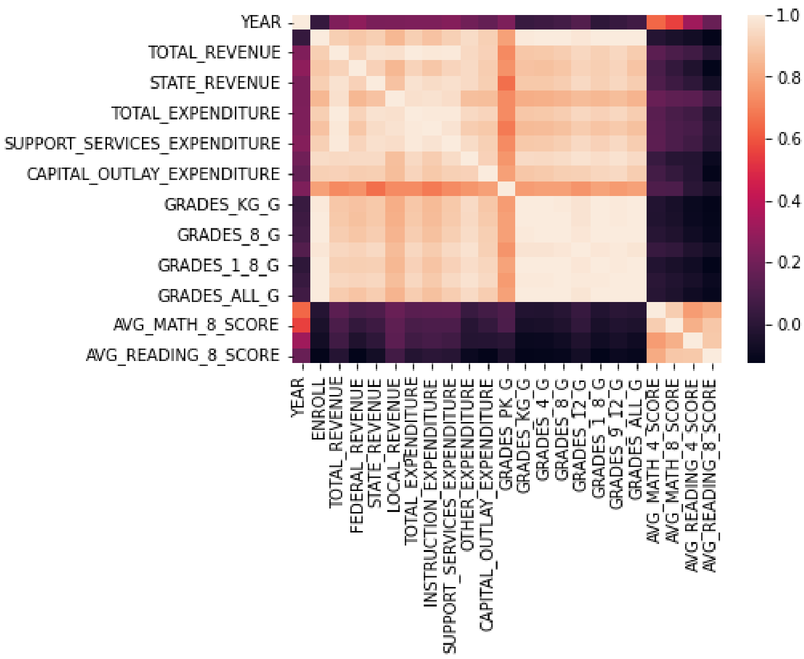
In [12]:
```
sns.pairplot(data)
```

Out[12]:<seaborn.axisgrid.PairGrid at 0x19f936db850>



In [13]:
```
sns.heatmap(data.corr())
```

Out[13]:<AxesSubplot:>



In [ ]: Уберем признаки, которые практически не влияют на другие признаки.

In [15]:
```
data.pop('GRADES_PK_G')
data.pop('GRADES_KG_G')
data.pop('GRADES_4_G')
data.pop('GRADES_8_G')
data.pop('GRADES_12_G')
data.pop('GRADES_1_8_G')
data.pop('GRADES_9_12_G')
data.pop('GRADES_ALL_G')
data.pop('AVG_MATH_4_SCORE')
data.pop('AVG_MATH_8_SCORE')
data.pop('AVG_READING_4_SCORE')
data.pop('AVG_READING_8_SCORE')
```

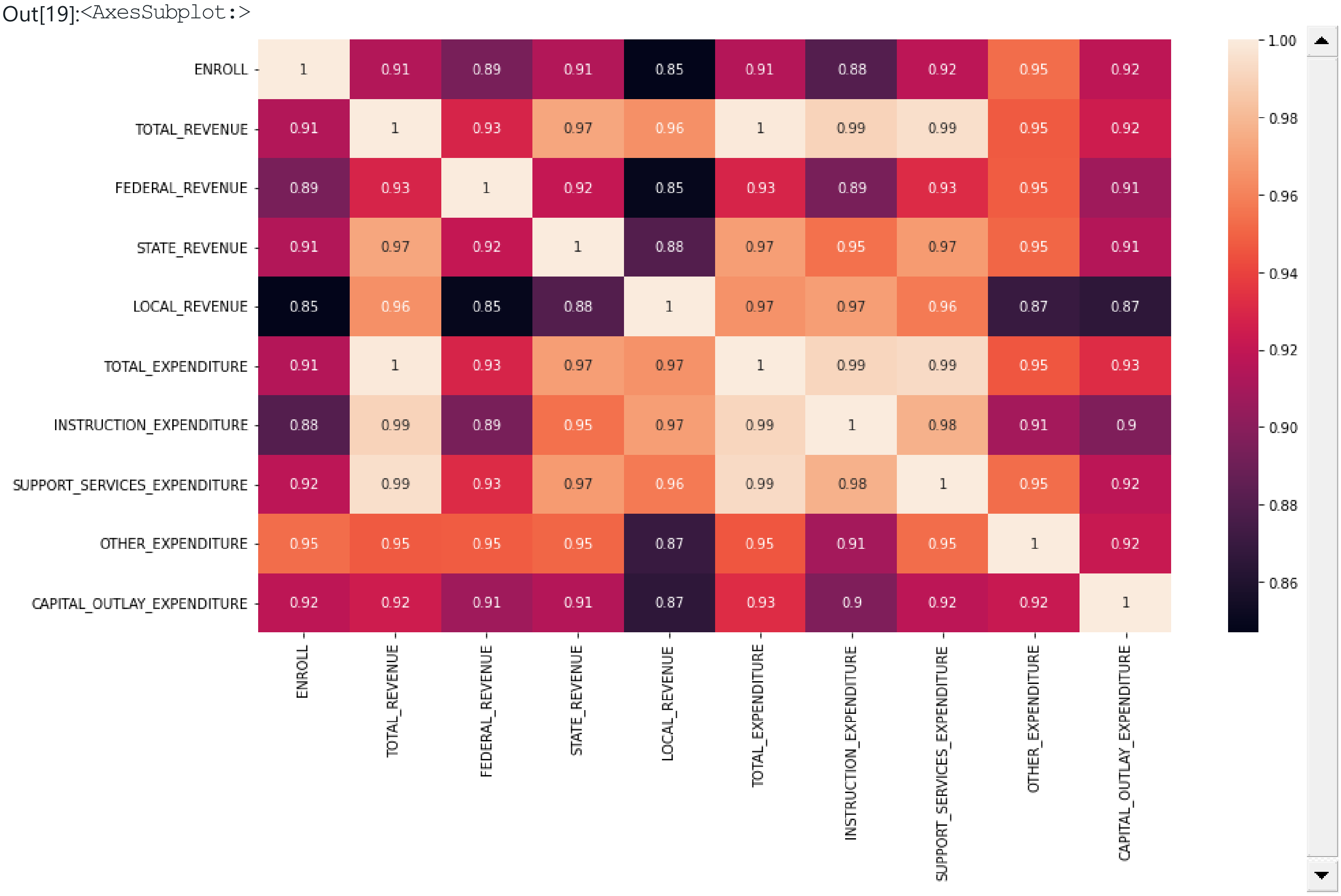
Out[15]:
```
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
        ...
1710    262.0
1711    266.0
1712    256.0
1713    267.0
1714    265.0
Name: AVG_READING_8_SCORE, Length: 1715, dtype: float64
```

In [16]:
```
data.pop('YEAR')
```

Out[16]:
```
0       1992
1       1992
2       1992
3       1992
4       1992
        ...
1710    2019
1711    2019
1712    2019
1713    2019
1714    2019
Name: YEAR, Length: 1715, dtype: int64
```

In [19]:
```
plt.figure(figsize = (15,8))
sns.heatmap(data.corr(), annot=True)
```

Out[19]:<AxesSubplot:>



Если брать целевым признак "enroll", то можно выбрать признаки, хорошо коррелирующие с ним, например "OTHER_EXPENDITURE ", "CAPITAL_OUTLAY_EXPENDITURE".