In [1]:

```
import numpy as np
import pandas as pd
data = pd.read_csv('C:\\states_all.csv')
```

In [2]:

```
data
```

Out[2]:

	PRIMARY_KEY	STATE	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPEND
0	1992_ALABAMA	ALABAMA	1992	NaN	2678885.0	304177.0	1659028.0	715680.0	265
1	1992_ALASKA	ALASKA	1992	NaN	1049591.0	106780.0	720711.0	222100.0	97
2	1992_ARIZONA	ARIZONA	1992	NaN	3258079.0	297888.0	1369815.0	1590376.0	340
3	1992_ARKANSAS	ARKANSAS	1992	NaN	1711959.0	178571.0	958785.0	574603.0	174
4	1992_CALIFORNIA	CALIFORNIA	1992	NaN	26260025.0	2072470.0	16546514.0	7641041.0	2713
...	...	...	...	...	...	...	...	...	
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 [22]:

```
data.columns
```

Out[22]:

```
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 [3]:

```
data.shape
```

Out[3]:

```
(1715, 25)
```

(в наборе данных 1715 строк и 25 столбцов)

In [4]:

```
data.dtypes
```

Out[4]:

```
PRIMARY_KEY                     object
STATE                           object
YEAR                             int64
ENROLL                         float64
TOTAL_REVENUE                  float64
FEDERAL_REVENUE                float64
STATE_REVENUE                  float64
LOCAL_REVENUE                  float64
TOTAL_EXPENDITURE              float64
INSTRUCTION_EXPENDITURE        float64
SUPPORT_SERVICES_EXPENDITURE   float64
OTHER_EXPENDITURE              float64
CAPITAL_OUTLAY_EXPENDITURE     float64
GRADES_PK_G                    float64
GRADES_KG_G                    float64
GRADES_4_G                     float64
GRADES_8_G                     float64
GRADES_12_G                    float64
GRADES_1_8_G                   float64
GRADES_9_12_G                  float64
GRADES_ALL_G                   float64
AVG_MATH_4_SCORE               float64
AVG_MATH_8_SCORE               float64
AVG_READING_4_SCORE            float64
AVG_READING_8_SCORE            float64
dtype: object
```

In [5]:

```
data.isnull().sum()
```

Out[5]:

```
PRIMARY_KEY                      0
STATE                            0
YEAR                             0
ENROLL                         491
TOTAL_REVENUE                  440
FEDERAL_REVENUE                440
STATE_REVENUE                  440
LOCAL_REVENUE                  440
TOTAL_EXPENDITURE              440
INSTRUCTION_EXPENDITURE        440
SUPPORT_SERVICES_EXPENDITURE   440
OTHER_EXPENDITURE              491
CAPITAL_OUTLAY_EXPENDITURE     440
GRADES_PK_G                    173
GRADES_KG_G                     83
GRADES_4_G                      83
GRADES_8_G                      83
GRADES_12_G                     83
GRADES_1_8_G                   695
GRADES_9_12_G                  644
GRADES_ALL_G                    83
AVG_MATH_4_SCORE              1150
AVG_MATH_8_SCORE              1113
AVG_READING_4_SCORE           1065
AVG_READING_8_SCORE           1153
dtype: int64
```

In [26]:

```
1153/1715
```

Out[26]:

```
0.6723032069970846
```

Какие выводы мы можем сделать о данных и об их возможном вкладе в модель?

- Колонки GRADES_KG_G, GRADES_4_G, GRADES_8_G, GRADES_12_G, GRADES_ALL_G содержат менее 5% пропусков, их можно точно безопасно включать в модель и вообще строить её на их основе, то же справедливо и в отношении колонки GRADES_PK_G с примерно 10% пропущенных строк.
- Колонки ENROLL, TOTAL_REVENUE, FEDERAL_REVENUE, STATE_REVENUE, LOCAL_REVENUE, TOTAL_EXPENDITURE, INSTRUCTION_EXPENDITURE, SUPPORT_SERVICES_EXPENDITURE, OTHER_EXPENDITURE,CAPITAL_OUTLAY_EXPENDITURE содержат 25-28% пропущенных данных, их можно включать в модель.
- Колонки AVG_MATH_4_SCORE, AVG_MATH_8_SCORE, AVG_READING_4_SCORE, AVG_READING_8_SCORE содержат 62-67% пропусков - это слишком много, эти признаки лучше не стоит включать в модель.

Вывели количество пропусков и тип данных в каждой колонке. Теперь можно заполнять пропуски.

Заполнение пропусков

Количественные данные

Будем заполнять столбцы AVG_MATH_4_SCORE (средний балл по математике среди учеников 4 класса) и AVG_READING_4_SCORE (средний балл по чтению среди учеников 4 класса).

Я решил очистить от пропусков сразу два столбца, чтобы потом на их основе построить диаграмму jointplot, указанную в качестве доп.задания, и посмотреть, как соотносятся между собой эти признаки.

In [6]:
```
data.AVG_MATH_4_SCORE.nunique()
```

Out[6]:
```
50
```

In [7]:
```
data.AVG_MATH_4_SCORE.describe()
```

Out[7]:
```
count    565.000000
mean     236.327434
std        9.285505
min      192.000000
25%      232.000000
50%      238.000000
75%      242.000000
max      253.000000
Name: AVG_MATH_4_SCORE, dtype: float64
```

In [8]:
```
from sklearn.impute import SimpleImputer
```

In [9]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
```

In [10]:
```
data.AVG_MATH_4_SCORE = imputer.fit_transform(data[['AVG_MATH_4_SCORE']])
```

In [11]:
```
data.AVG_MATH_4_SCORE.isnull().any()
```

Out[11]:
```
False
```

In [12]:
```
data.AVG_MATH_4_SCORE.describe()
```

Out[12]:
```
count    1715.000000
mean      237.448980
std         5.384205
min       192.000000
25%       238.000000
50%       238.000000
75%       238.000000
max       253.000000
Name: AVG_MATH_4_SCORE, dtype: float64
```

В качестве "заглушки" для пропусков я использовал медиану, поэтому можно увидеть, как поменялось значение среднего по колонке.

In [13]:
```
data.AVG_READING_4_SCORE.isnull().sum()
```

Out[13]:
```
1065
```

In [14]:
```
data.AVG_READING_4_SCORE = imputer.fit_transform(data[['AVG_READING_4_SCORE']])
```

In [15]:
```
data.AVG_READING_4_SCORE.isnull().sum()
```

Out[15]:
```
0
```

Итак, оба столбца заполнены, можем построить для них график jointplot.

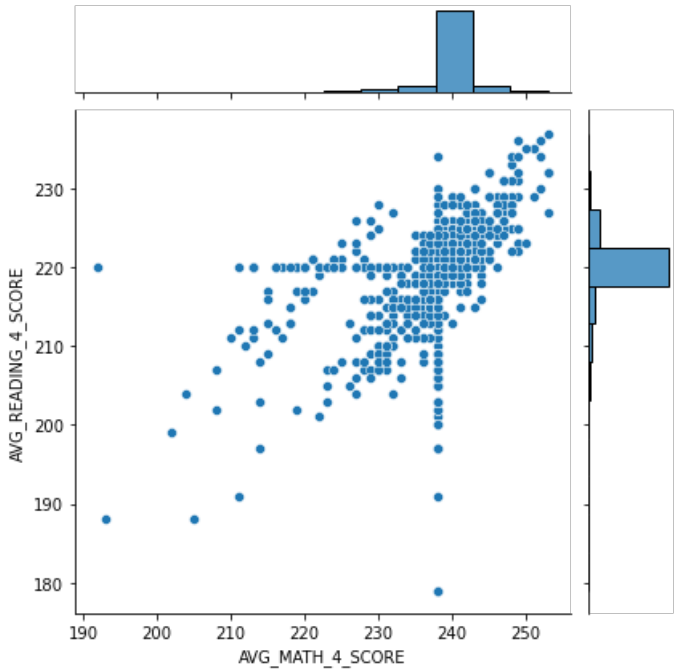
In [16]:
```
import seaborn as sns
```

In [18]:

```
sns.jointplot(data=data,x="AVG_MATH_4_SCORE",y="AVG_READING_4_SCORE")
```

Out[18]:

```
<seaborn.axisgrid.JointGrid at 0x1632937aaf0>
```



Категориальные данные

К сожалению, такое задание будет невозможно сделать на попавшем мне датасете, потому что есть только два признака с типом object: PRIMARY_KEY и STATE - и в обоих нет ни одного пропуска.