#df\_test = df\_test.merge(df\_full[['final.output.recovery', 'rougher.output.recovery', 'date']], how='left', on='date') #df.columns = df.columns.str.lower() #df.query('critic\_score.isna()').groupby('year\_of\_release')['na\_sales'].count() #df.dropna(subset=['year\_of\_release'], inplace=True) Приеобразование типа данных

df['user\_score'] = df['user\_score'].astype(float) df['user\_score'] = df['user\_score'].astype(float) Рейтинг K-A - старое название E, поэтому их необходимо будет объединить

df['rating'] = df['rating'].replace('K-A', 'E') df['rating'] = df['rating'].replace('K-A', 'E')

**Создание заглушек для rating** df.loc[df['rating'].isna(),'rating'] = 'U'

Суммарные продажи

sum\_total = data.groupby('platform').agg(total\_sales=('total\_sales','sum')).sort\_values(by='total\_sales', ascending = False)

Постоение распределения по годам sales\_year\_distr = data.pivot\_table(index=['family'], columns=['year\_of\_release'], values='total\_sales', aggfunc=['sum']).droplevel(level=0,axis=1)

заменим е на ё и все посёлки, кроме городского типа преобразуем просто в посёлки

df.replace('(поселок)','посёлок',regex=True, inplace = True) df.replace('(городской посёлок)','посёлок городского типа',regex=True, inplace = True) df.replace('(поселок)','посёлок',regex=True, inplace = True) df.replace('(городской посёлок)','посёлок городского типа',regex=True, inplace = True) df.replace('(посёлок городского типа)','пгт.',regex=True, inplace = True) df.replace('(посёлок при железнодорожной станции|коттеджный посёлок)','посёлок',regex=True, inplace = True) df.replace('(посёлок городского типа)','пгт.',regex=True, inplace = True) df.replace('(посёлок городского типа)','пгт.',regex=True, inplace = True)

Наименования столбцов не по канону, меняем на правильные

df.columns = df.columns.map(lambda x: re.sub( '(?<!^)(?=[A-Z])', '', x ).lower()) df.columns = df.columns.map(lambda x: re.sub( '(?<!^)(?=[A-Z])', '', x ).lower())

**------------------**

**Applying function with multiple arguments to create a new pandas column**

>>> def fxy(x, y):

... return x \* y

>>> df['newcolumn'] = df.apply(lambda x: fxy(x['A'], x['B']), axis=1)

>>> df

A B newcolumn

0 10 20 200

1 20 30 600

2 30 10 300

--------------------------------------