In [2]:
```
from sklearn.neural_network  import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

فراخوانی کتابخانه های مورد نیاز برای ساخت شبکه عصبی و معیار های سنجش شبکه عصبی ساخته شده

In [3]:
```
import joblib as jl
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', 50000)
pd.set_option('display.max_columns', 500)
#Note that .set_option() changes behavior globaly in Jupyter Notebooks, so i
pd.set_option('display.float_format', lambda x: '%.5f' % x)
#FOR NUMPY
np.set_printoptions(suppress=True)
```

فراخوانی کتابخانه های لازم برای رسم نمودار و کار با داده ها

In [4]:
```
DDD_DataSet=jl.load('DDD_DiD_DataSet.pkl')
DDD_DataSet.head()
```

Out[4]:

	year	month	fcn	fcnv_DDD	fcnv_DID
0	2011	1	J01AA	808432.00000	0.17064
1	2011	2	J01AA	616620.00000	0.13945
2	2011	3	J01AA	802755.00000	0.16945
3	2011	4	J01AA	736725.66700	0.15551
4	2011	5	J01AA	1038556.00000	0.21922

اپلود کردن فایل داده های خروجی فاز دو

In [18]:
```
DDD_DataSet.isna().sum()
```

Out[18]:
```
year        0
month       0
fcn         0
fcnv_DDD    0
fcnv_DID    0
dtype: int64
```

داده های گم شده را در فاز دو با مقادیر قبلی جایگذین کردیم

In [7]:
```
DDD_DataSet.describe()
```

Out[7]:

	year	month	fcnv_DDD	fcnv_DID
count	1860.00000	1860.00000	1860.00000	1860.00000
mean	2013.00000	6.50000	1230788.82416	0.28555
std	1.41459	3.45298	1980714.63241	0.46169
min	2011.00000	1.00000	0.00000	0.00000
25%	2012.00000	3.75000	36613.71600	0.00818
50%	2013.00000	6.50000	153953.54000	0.03644
75%	2014.00000	9.25000	1388765.12800	0.32230
max	2015.00000	12.00000	12492330.34000	3.07244

خلاصه اطلاعات آماری داده ها

In [8]:
```
DDD_DataSet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1860 entries, 0 to 1859
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   year      1860 non-null   int64  
 1   month     1860 non-null   int64  
 2   fcn       1860 non-null   object 
 3   fcnv_DDD  1860 non-null   float64
 4   fcnv_DID  1860 non-null   float64
dtypes: float64(2), int64(2), object(1)
memory usage: 72.8+ KB
```

In [10]:
```
#fcn::  fine class name for antibitic

def data_per_fca(fcn):
    DDD_DataSet=jl.load('DDD_DiD_DataSet.pkl')
    print('fcn:: ',fcn)
    x=DDD_DataSet.fcnv_DDD[DDD_DataSet['fcn']==fcn]
    x.reset_index(inplace=True,drop=True)
    print('data_per_',fcn,' shape::',x.shape)
    return x
```

در این بخش طبق تابع ساخته شده پیش بینی برای هر گروه از آنتی بیوتیک به طور جداگانه انجام خواهد شد و کار این تابع دریافت گروه مشخصی از آنتی بیوتیک برای قرار گرفتن در شبکه عصبی می باشد.

In [11]:
```
def DDD_Data_generate_byhistory(data_per_fca_,fcn,periods_,):
    #Create Historical data-----------------------------------------------------
    data_per_fca_byhistory=pd.DataFrame(data_per_fca_)
    columns_name=['fcnv_DDD']
    print(data_per_fca_byhistory.shape)
    prd=0

    for prd in range(1,periods_+1):
        data_per_fca_byhistory= pd.concat(
            [   data_per_fca_byhistory,
                data_per_fca_.shift(periods=prd,fill_value=0)
            ],axis=1
             )
        print(data_per_fca_byhistory.shape)
        columns_name.append('fcnv_DDD_shift'+str(prd))
        data_per_fca_byhistory.columns=columns_name

        data_per_fca_byhistory.reset_index(inplace=True,drop=True)

    #End of for-----------------------------------------------------------------
    return data_per_fca_byhistory
```

کاری که این تابع انجام خواهد داد طبق تابع تعریف شده قبلی و اسم گروه آنتی بیوتیک و همچنین دوره مشخص که آرگومان های ورودی هستند برای فایل داده اصلی تاریخچه ایی ایجاد خواهد کرد که در فاز دو به طور کامل ضرورت ایجاد تاریخچه توضیح داده شده است

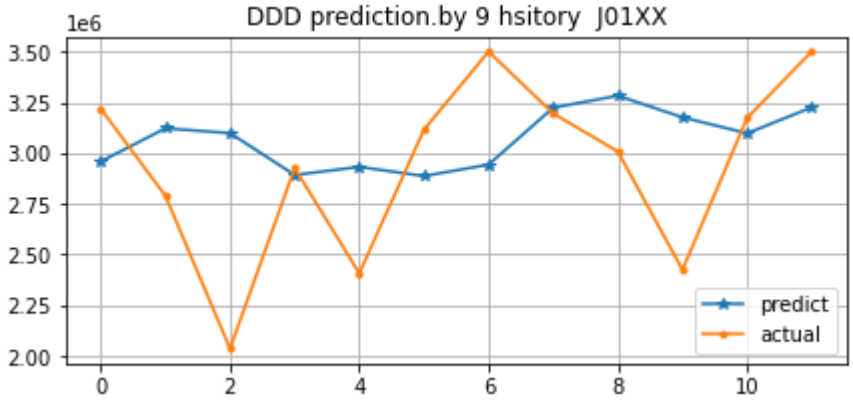
In [19]:
```
DDD_DataSet.fcn.unique()
```

Out[19]:
```
array(['J01AA', 'J01BA', 'J01CA', 'J01CE', 'J01CF', 'J01CG', 'J01CR',
       'J01DA', 'J01DB', 'J01DC', 'J01DD', 'J01DE', 'J01DF', 'J01DH',
       'J01DI', 'J01EA', 'J01EB', 'J01EC', 'J01EE', 'J01FA', 'J01FF',
       'J01FG', 'J01GA', 'J01GB', 'J01MA', 'J01XA', 'J01XB', 'J01XC',
       'J01XD', 'J01XE', 'J01XX'], dtype=object)
```

In [15]:
```
periods_=np.random.randint(2,12)
periods_=9
fcn='J01XX'
data_per_fca(fcn)
DDD_data_byhistory=DDD_Data_generate_byhistory(data_per_fca(fcn),fcn,periods


Y=DDD_data_byhistory.loc[:,['fcnv_DDD']].values.ravel()
X=DDD_data_byhistory.iloc[:,1:periods_]
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
shfl=False
rnd_std=444
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,shuffle=shi
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
hlayer_1_size=60
hlayer_2_size=17
#print('hidden layer size:',(hlayer_1_size,hlayer_2_size))
model_DDD=MLPRegressor((hlayer_1_size,hlayer_2_size),max_iter=5000,verbose=I
model_DDD.fit(X_train,Y_train)
y_pred=model_DDD.predict(X_test)
jl.dump(model_DDD,'model_DDD.pkl')
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
RMSE=round(mean_squared_error(Y_test,y_pred,squared=False) ,2)
MSE=round(mean_squared_error(Y_test,y_pred),2)
MAE=round(mean_absolute_error(Y_test,y_pred),2)
print('rmse:'.upper(),RMSE)
print('mse:'.upper(),MSE)
print('mae:'.upper(),MAE)
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
plt.figure(figsize=(7,3))
plt.title('DDD prediction.by '+str(periods_)+' hsitory  '+fcn)
plt.plot(y_pred,ls='-',marker='*')
plt.plot(Y_test,ls='-',marker='.')
plt.legend(['predict','actual'])
plt.grid()
plt.show()
```

```
fcn::  J01XX
data_per_ J01XX  shape:: (60,)
fcn::  J01XX
data_per_ J01XX  shape:: (60,)
(60, 1)
(60, 2)
(60, 3)
(60, 4)
(60, 5)
(60, 6)
```

```
(60, 7)
(60, 8)
(60, 9)
(60, 10)
RMSE: 471580.68
MSE: 222388336850.8
MAE: 367179.88
```



در این بخش با استفاده از تابع رندوم دوره ایی که برای ایجاد تاریخچه استفاده میشود ۹ است یعنی از ۹ ماه قبل استفاده میشود برای ایجاد تاریخچه و اسم گرروه خاصی که قرار است پیش بینی برای ان انجام شود انتخاب شده و دو تابع ایجاد شده فراخوانی شده اند و از این توابع در ساخت مدل شبکه عصبی استفاده شده است و شبکه عصبی ایجاد شده دارای دو لایه نهان با نورون های ۶۰ و ۱۷ میباشد و داده های تست و اموزش را ایجاد میکنیم و همچنین با استفاده از کتابخانه مربوط به ساخت شبکه عصبی مدل خودمان را ساخته و خروجی مدل را نیز ذخیره خواهیم کرد تا هرجا لازم شد از ان استفاده کنیم از طرفی از معیار های سنجش با استفاده از کتابخانه های مربوطه استفاده کرده و نمودار مربوط به مقدار واقعی و مقدار پیش بینی شده در شبکه عصبی را رسم میکنیم که مقدار واقعی در نمودار با نقطه و مقدار پیش بینی شده با ستاره مشخص شده است و البته باید توجه کرد که برای هر گروه آنتی بیوتیک معیار های سنجش و نمودار متفاوت خواهد شد

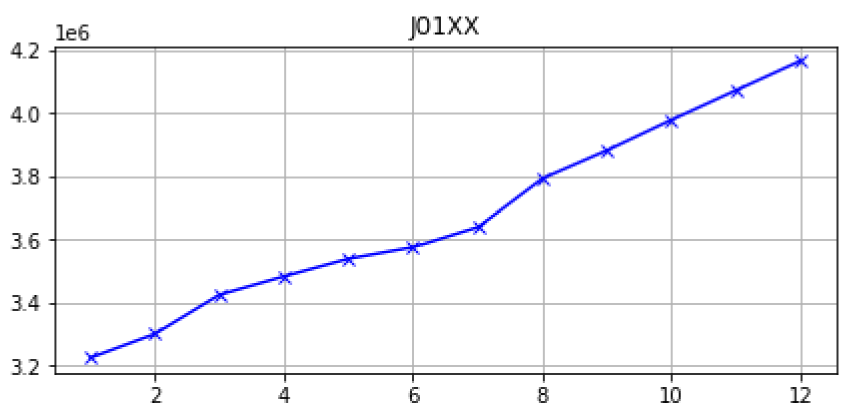
In [21]:
```
def predict_one_future_month(model, input_, periods_, num_month_next):
    global table_next_month
    if   num_month_next == 0 :
        return table_next_month
    one_future_month=model.predict(input_)
    #print(input_,input_.shape)
    added_one_future_month=np.concatenate(([one_future_month],input_),axis=1
    X_for_next_month=added_one_future_month[0,0:periods_-1].reshape(1,-1)

    #print(X_for_next_month,X_for_next_month.shape)

    table_next_month=np.concatenate((table_next_month,X_for_next_month))
    predict_one_future_month(model,X_for_next_month,periods_,num_month_next-
```

با استفاده از این تابع میتوان هر تعداد ماه از اینده را که میخواهیم برای گروهی خاصی از آنتی بیوتیک پیش بینی کنیم

In [24]:
```
model=jl.load('model_DDD.pkl')
num_month_next=12
input_=X.iloc[59,:].values.reshape(1,-1)
table_next_month=np.array([np.ones(periods_-1)])
predict_one_future_month(model,input_,periods_,num_month_next)

#print(table_next_month.shape)
print(table_next_month)
print(table_next_month[1:num_month_next+1,0])
plt.figure(figsize=(7,3))
plt.plot(range(1,num_month_next+1),table_next_month[1:num_month_next+1,0],'k

plt.title(fcn)
plt.grid()
plt.show()
```

```
[[      1.              1.              1.              1.
        1.              1.              1.              1.        ]
 [3225825.54795217 3173443.283      2425091.744      3008060.505
  3195654.284      3500718.275      3115474.269      2407700.519     ]
 [3301372.55786922 3225825.54795217 3173443.283      2425091.744
  3008060.505      3195654.284      3500718.275      3115474.269     ]
 [3425213.18124155 3301372.55786922 3225825.54795217 3173443.283
  2425091.744      3008060.505      3195654.284      3500718.275     ]
 [3483364.09671127 3425213.18124155 3301372.55786922 3225825.54795217
  3173443.283      2425091.744      3008060.505      3195654.284     ]
 [3539233.62079894 3483364.09671127 3425213.18124155 3301372.55786922
  3225825.54795217 3173443.283      2425091.744      3008060.505     ]
 [3575952.69545625 3539233.62079894 3483364.09671127 3425213.18124155
  3301372.55786922 3225825.54795217 3173443.283      2425091.744     ]
 [3639236.46611607 3575952.69545625 3539233.62079894 3483364.09671127
  3425213.18124155 3301372.55786922 3225825.54795217 3173443.283     ]
 [3792470.55946267 3639236.46611607 3575952.69545625 3539233.62079894
  3483364.09671127 3425213.18124155 3301372.55786922 3225825.54795217]
 [3881227.581791   3792470.55946267 3639236.46611607 3575952.69545625
  3539233.62079894 3483364.09671127 3425213.18124155 3301372.55786922]
 [3977703.86649116 3881227.581791   3792470.55946267 3639236.46611607
  3575952.69545625 3539233.62079894 3483364.09671127 3425213.18124155]
 [4071234.81580241 3977703.86649116 3881227.581791   3792470.55946267
  3639236.46611607 3575952.69545625 3539233.62079894 3483364.09671127]
 [4164506.81319469 4071234.81580241 3977703.86649116 3881227.581791
  3792470.55946267 3639236.46611607 3575952.69545625 3539233.62079894]]
[3225825.54795217 3301372.55786922 3425213.18124155 3483364.09671127
 3539233.62079894 3575952.69545625 3639236.46611607 3792470.55946267
 3881227.581791   3977703.86649116 4071234.81580241 4164506.81319469]

c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
```

```
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
c:\Users\ASUS\zahra\simple-project\env\lib\site-packages\sklearn\base.
py:445: UserWarning: X does not have valid feature names, but MLPRegre
ssor was fitted with feature names
  warnings.warn(
```



طبق این بخش از کد توانستیم مصرف آنتی بیوتیک را برای یکسال آینده پیش بینی کنیم و بر اساس نمودار متوجه میشویم که مصرف آنتی بیوتیک روند افزایشی داشته است