In [20]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [21]:
```
train_data=pd.read_csv('train.csv')
```

In [22]:
```
train_data.head()
```

Out[22]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X3
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	

5 rows × 378 columns

In [42]:
```
test_data=pd.read_csv('test.csv')
```

In [43]:
```
test_data.head()
```

Out[43]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X38
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	

5 rows × 377 columns

In [23]:
```
train_data.dtypes.value_counts()
```

Out[23]:
```
int64      369
object       8
float64      1
dtype: int64
```

Check for null and unique values for test and train sets

In [148…
```
null_cols_train=[]
for i in train_data.columns:
    if train_data[i].isna().sum()>0:
        null_cols_train.append(i)
```

In [149…
```
null_cols_test=[]
for i in test_data.columns:
    if test_data[i].isna().sum()>0:
        null_cols_test.append(i)
```

In [159…
```
unique_cols_train=[]
for i in train_data.columns:
    if len(train_data[i].unique())==1:
        unique_cols_train.append(i)
```

In [162…
```
unique_cols_test=[]
for i in test_data.columns:
    if len(test_data[i].unique())==1:
        unique_cols_test.append(i)
```

In [147…
```
null_cols_train
```

Out[147… []

In [146…
```
null_cols_test
```

Out[146… []

In [161…
```
unique_cols_train
```

Out[161…
```
['X11',
 'X93',
 'X107',
 'X233',
 'X235',
 'X268',
 'X289',
 'X290',
 'X293',
 'X297',
 'X330',
 'X347']
```

In [163…
```
unique_cols_test
```

Out[163… ['X257', 'X258', 'X295', 'X296', 'X369']

In [27]:
```
cat_cols=[]
binary_cols=[]
for col in train_data.columns:
    if train_data[col].dtype=='object':
        cat_cols.append(col)
    elif train_data[col].dtype=='int64' and col!='ID':
        binary_cols.append(col)
```

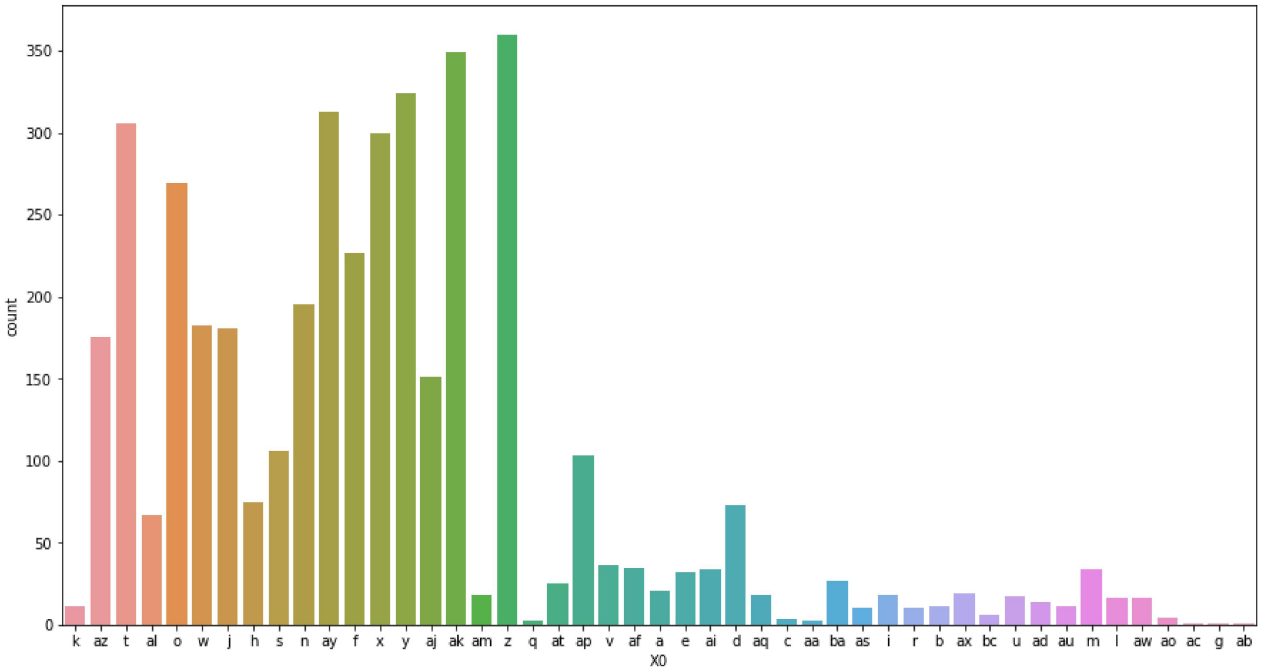
In [28]:
```
cat_cols
```

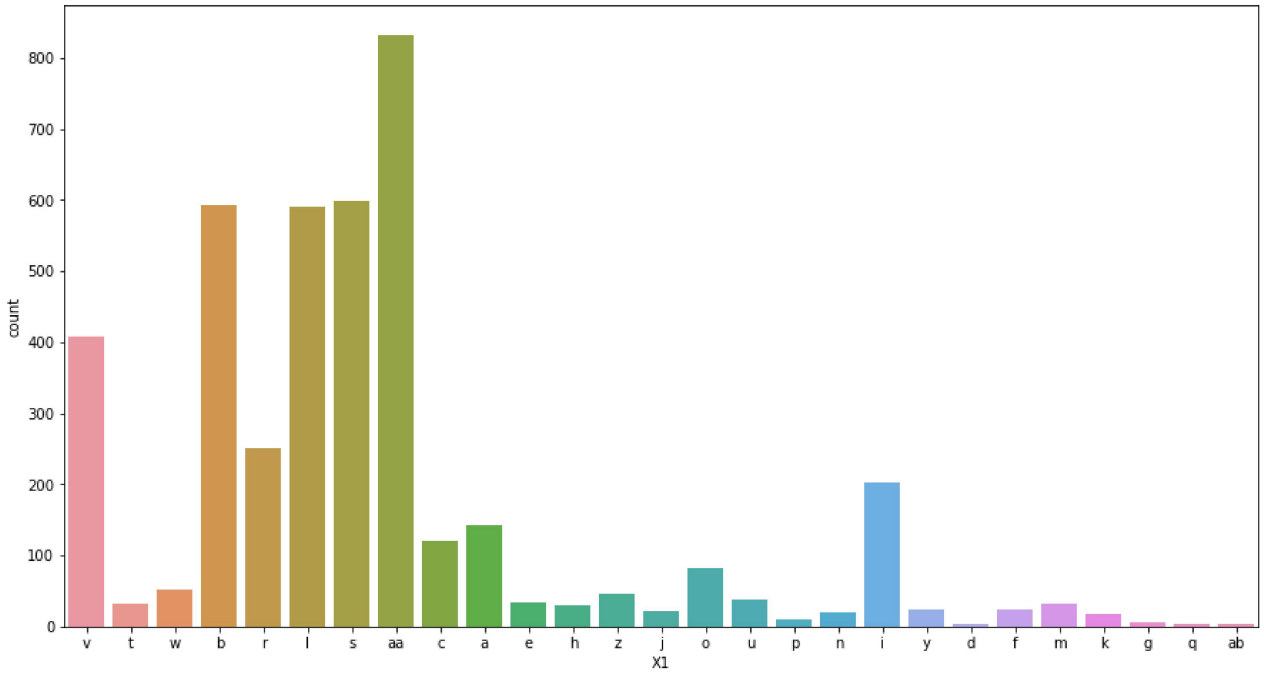
Out[28]:
```
['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

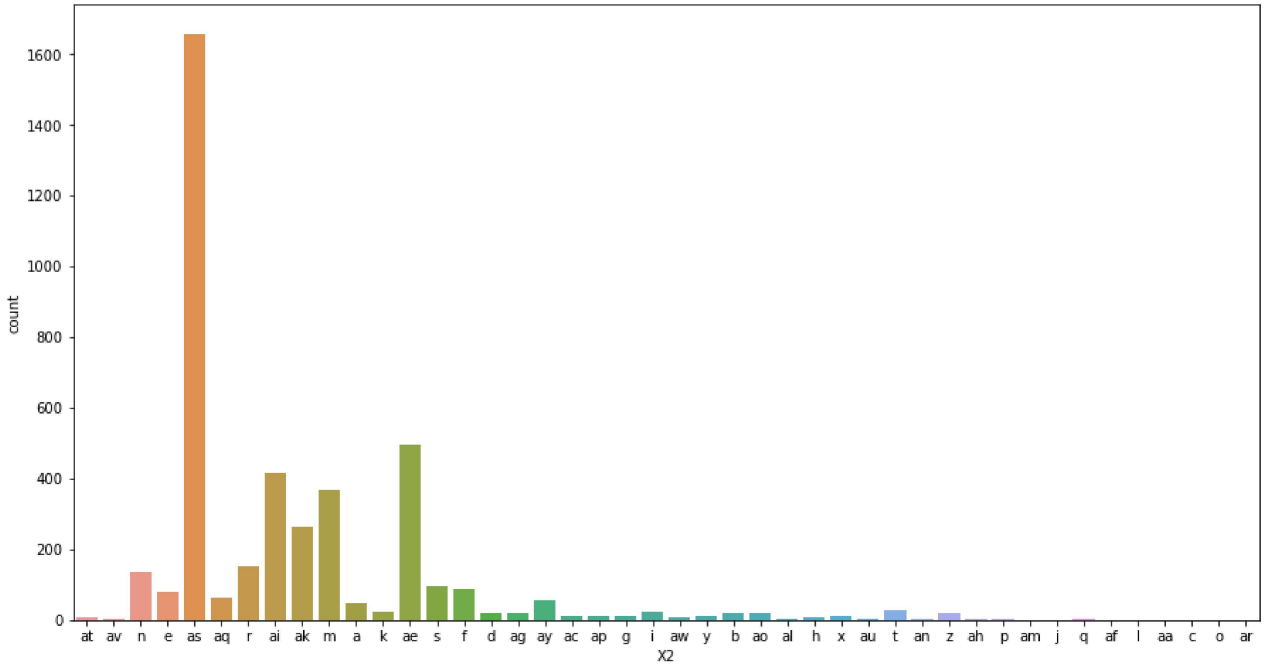
In [ ]:
```
binary_cols
```

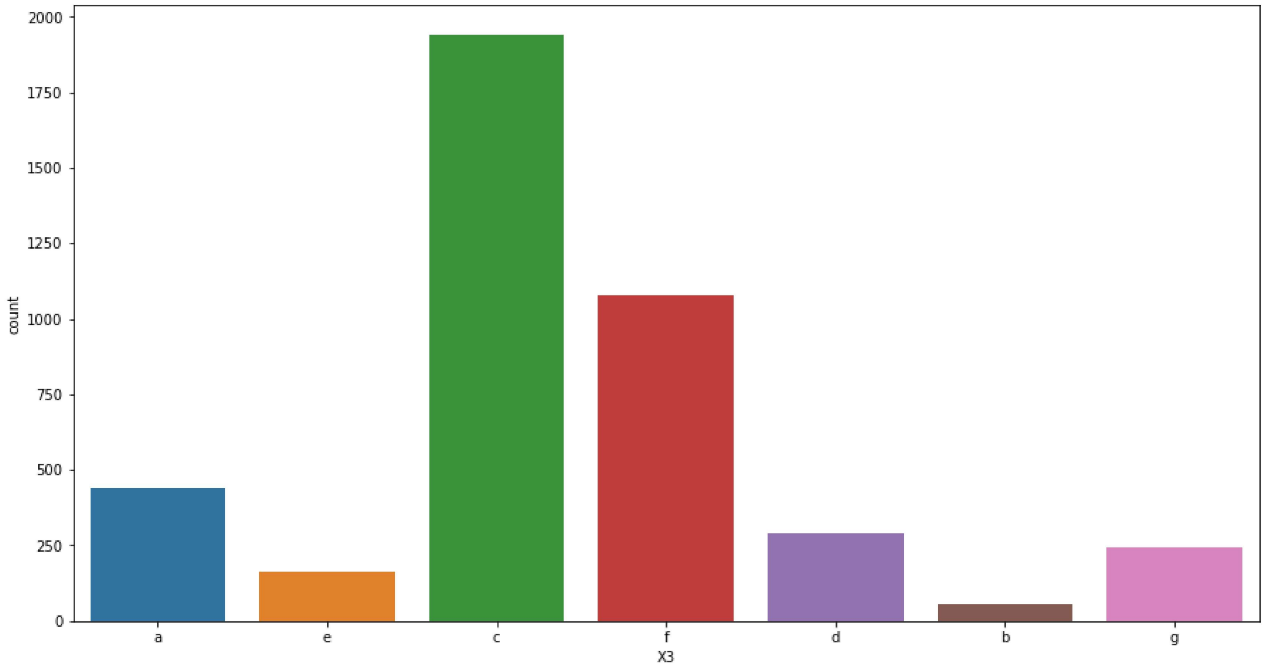
SOME QUICK EDA

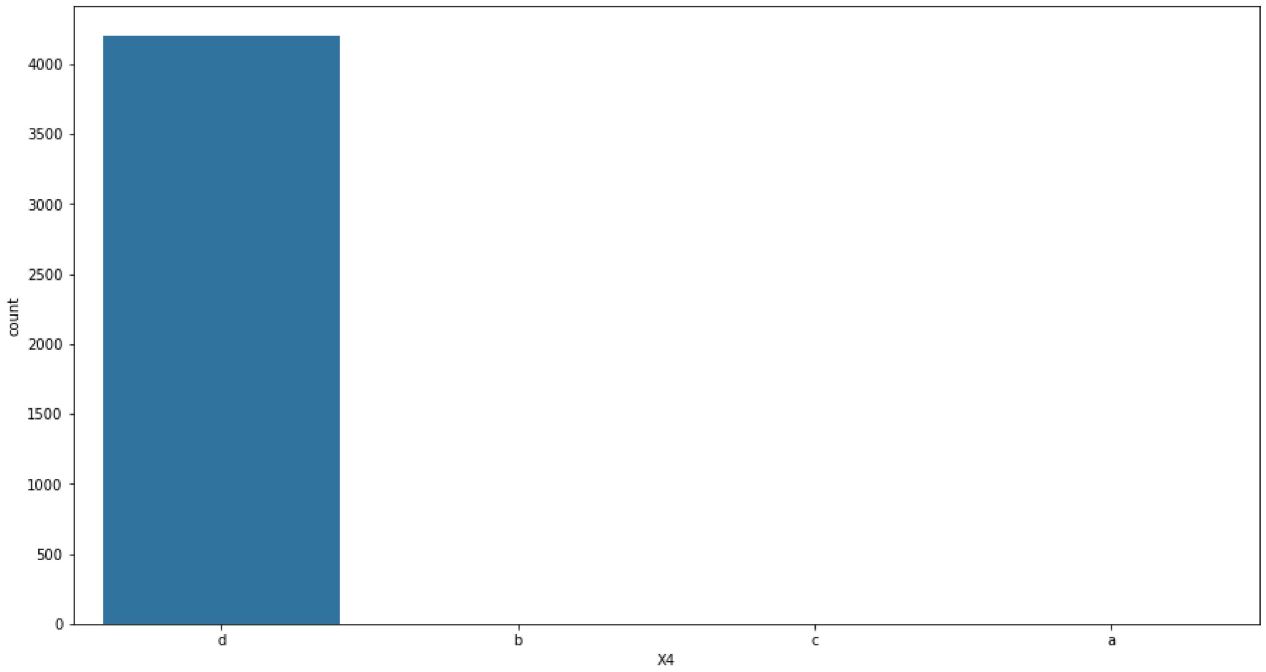
In [10]:
```
for col in cat_cols:
    plt.figure(figsize=(15,8))
    sns.countplot(x=col, data=train_data)
    plt.show()
#train_data[cat_cols].value_counts().plot(kind='barh', subplots=True)
```

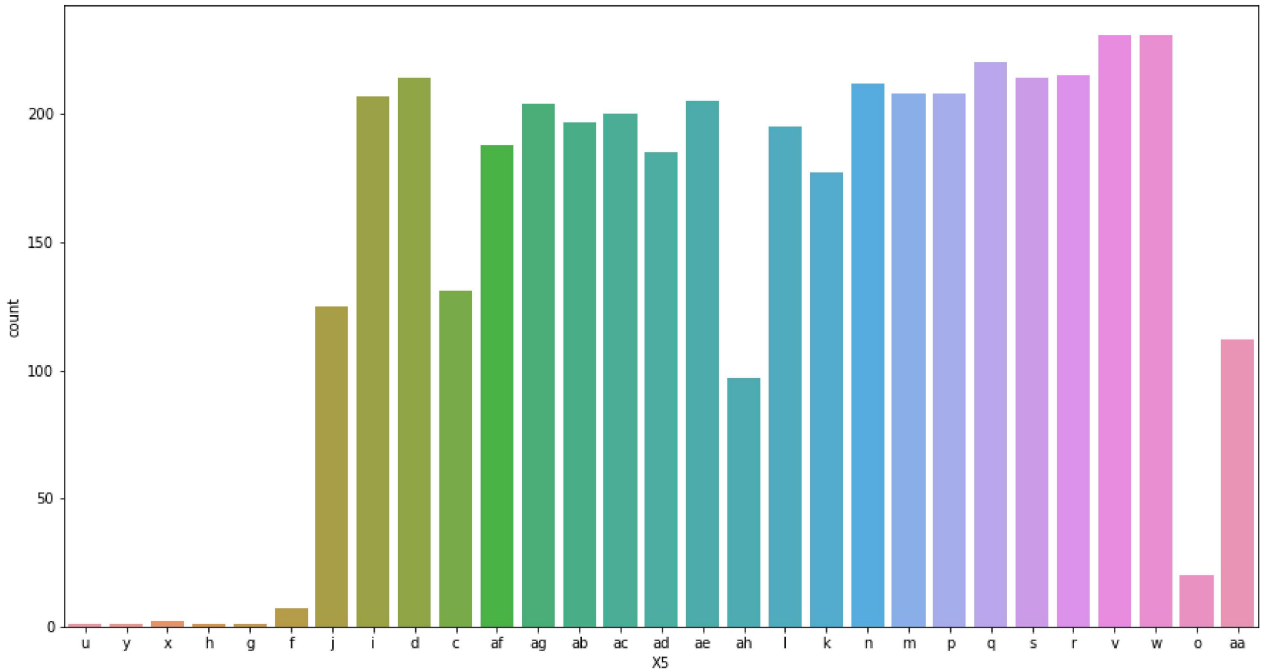


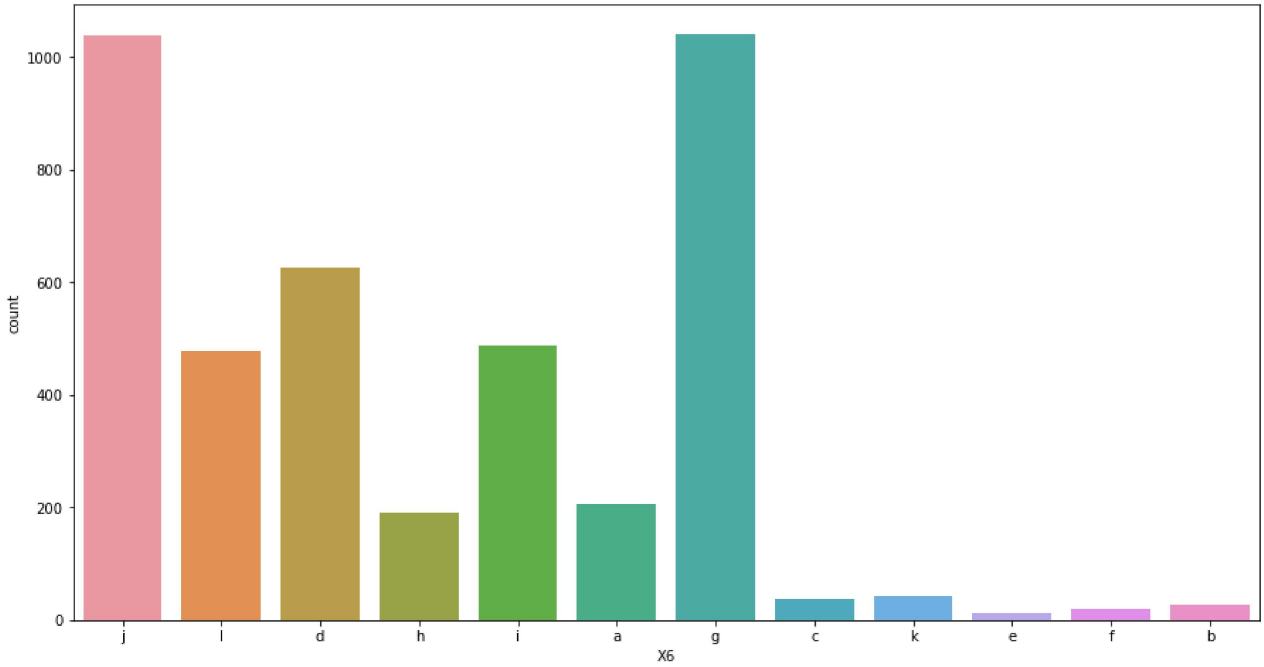

count
X1
v t w b r l s aa c a e h z j o u p n i y d f m k g q ab

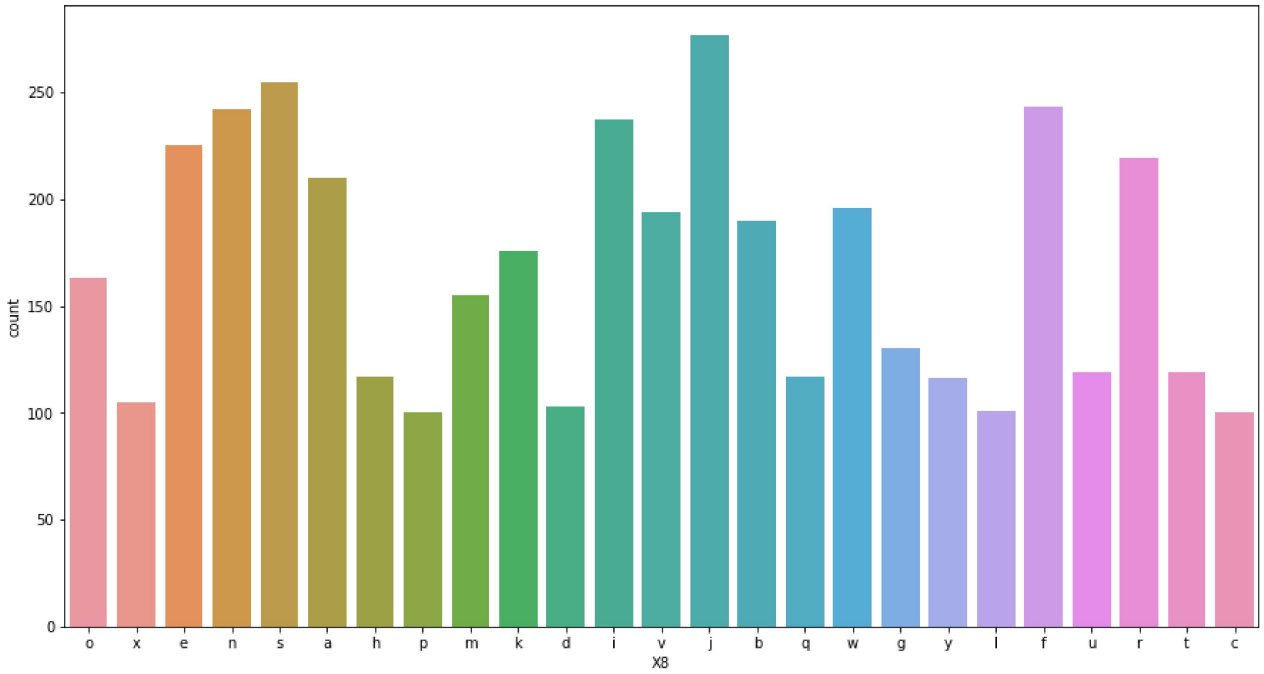

count
X2
at av n e as aq r ai ak m a k ae s f d ag ay ac ap g i aw y b ao al h x au t an z ah p am j q af l aa c o ar


2000
1750
1500
1250
1000
750
500
250
0
count
a
e
c
f
d
b
g
X3


4000
3500
3000
2500
2000
1500
1000
500
0
count
d
b
c
a
X4

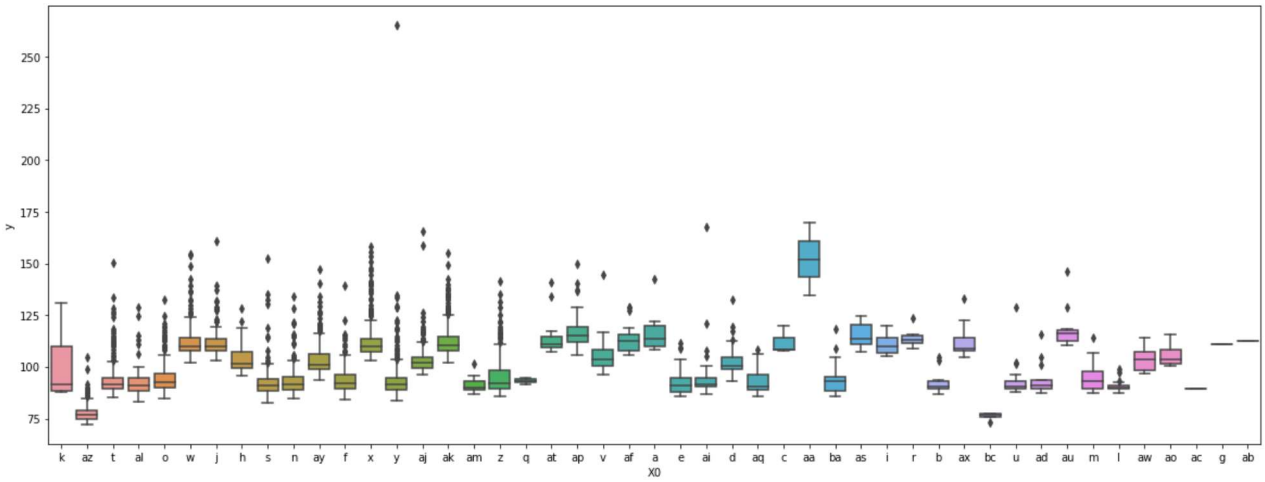

count
X5
u y x h g f j i d c af ag ab ac ad ae ah l k n m p q s r v w o aa
0 50 100 150 200

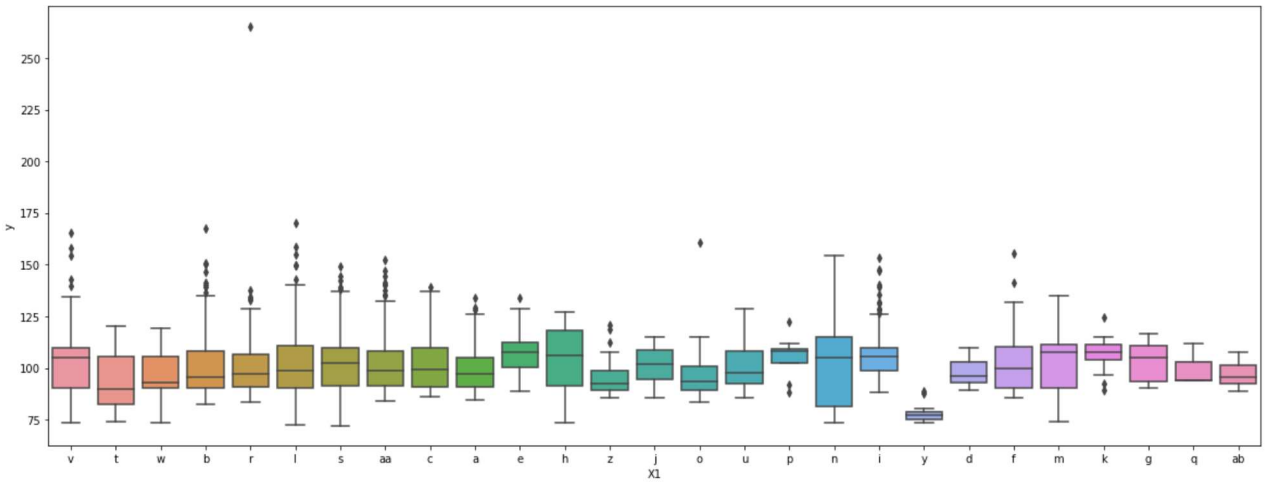

count
X6
j l d h i a g c k e f b
0 200 400 600 800 1000

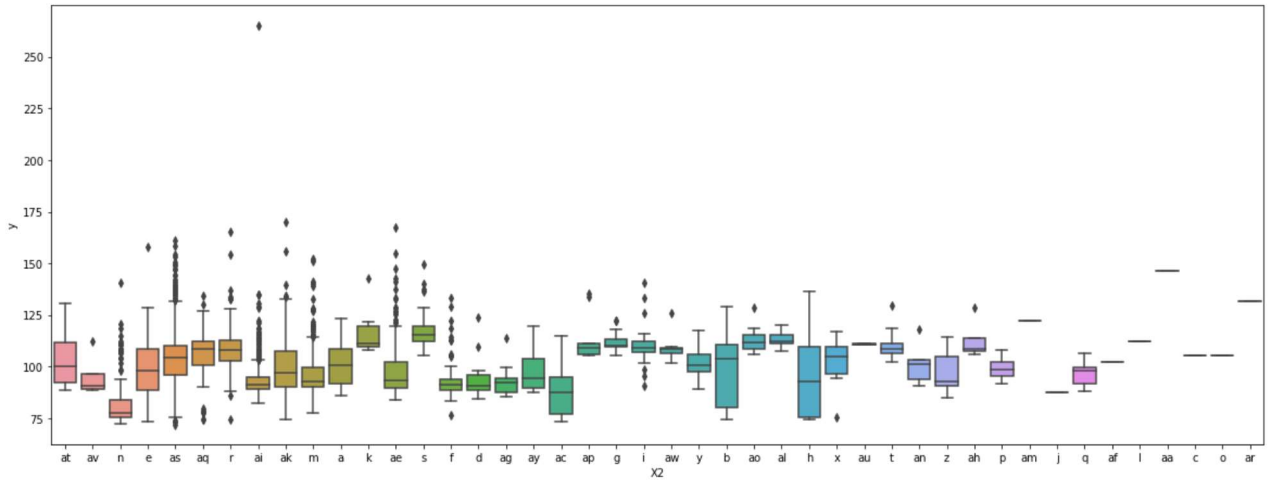


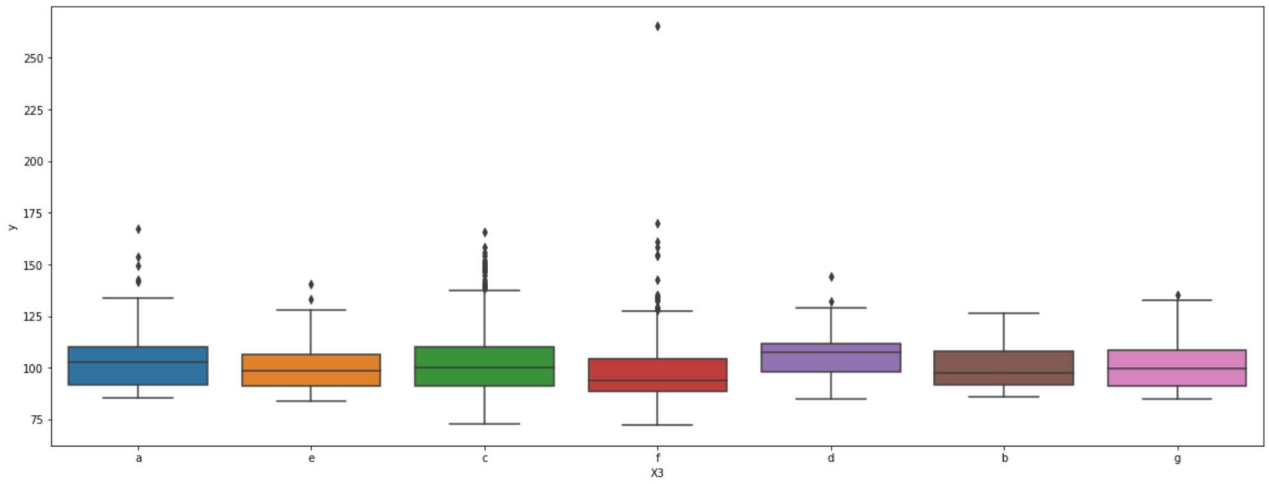
In [11]:

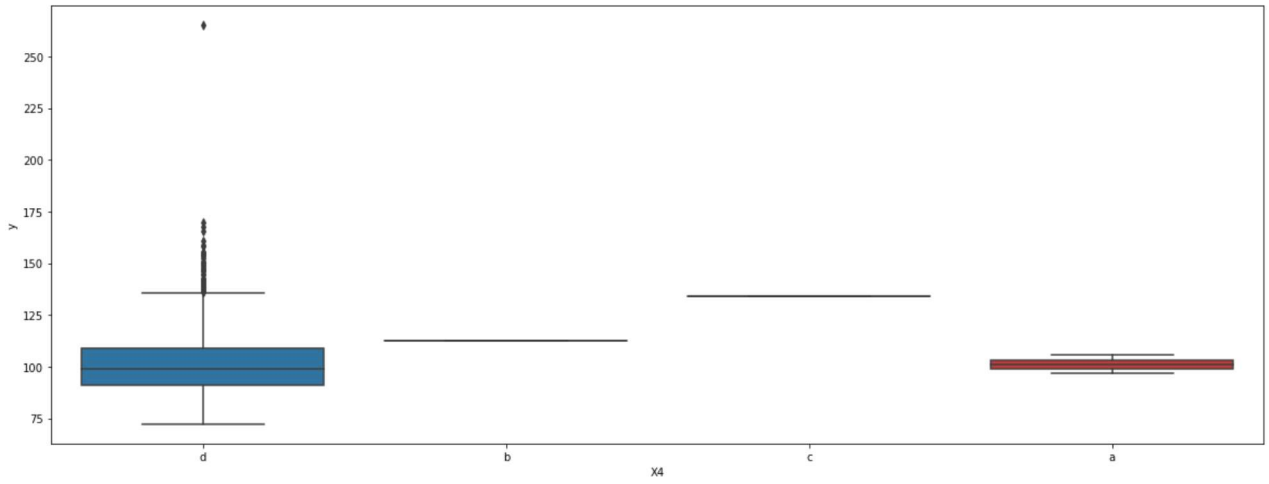
```
fig,ax = plt.subplots(len(cat_cols), figsize=(20,70))

for i, col in enumerate(cat_cols):
    sns.boxplot(x=col, y='y', data=train_data, ax=ax[i])
```

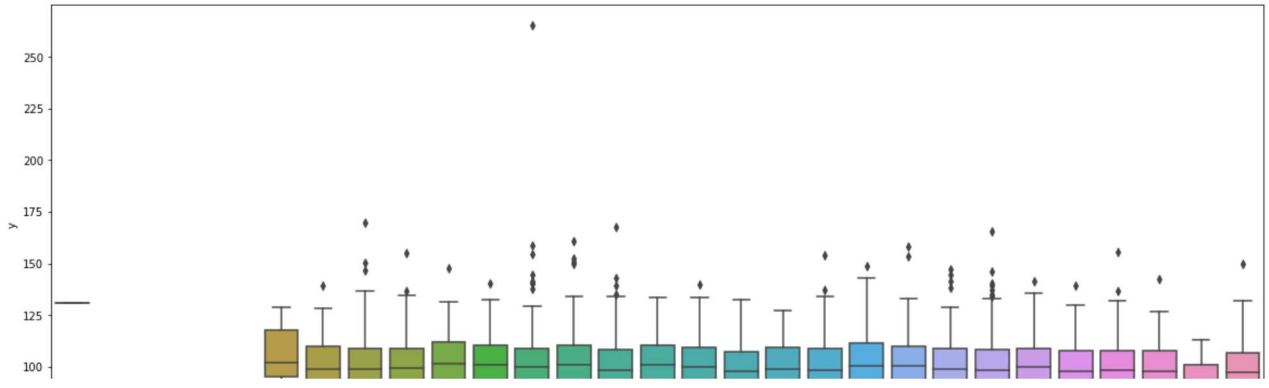


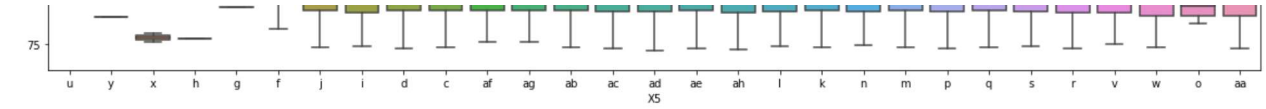


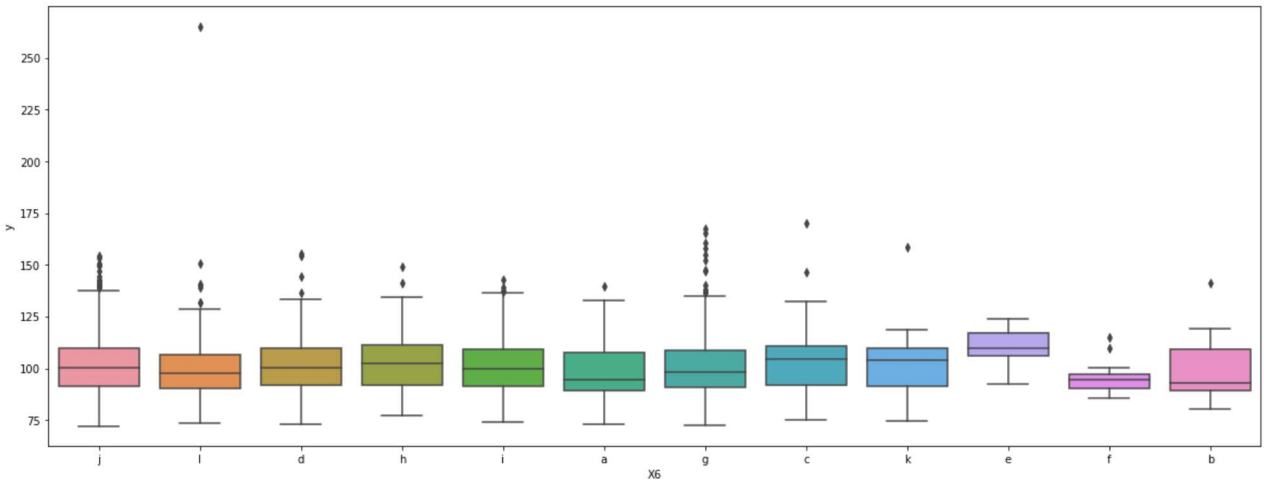

y
X2
at av n e as aq r ai ak m a k ae s f d ag ay ac ap g i aw y b ao al h x au t an z ah p am j q af l aa c o ar

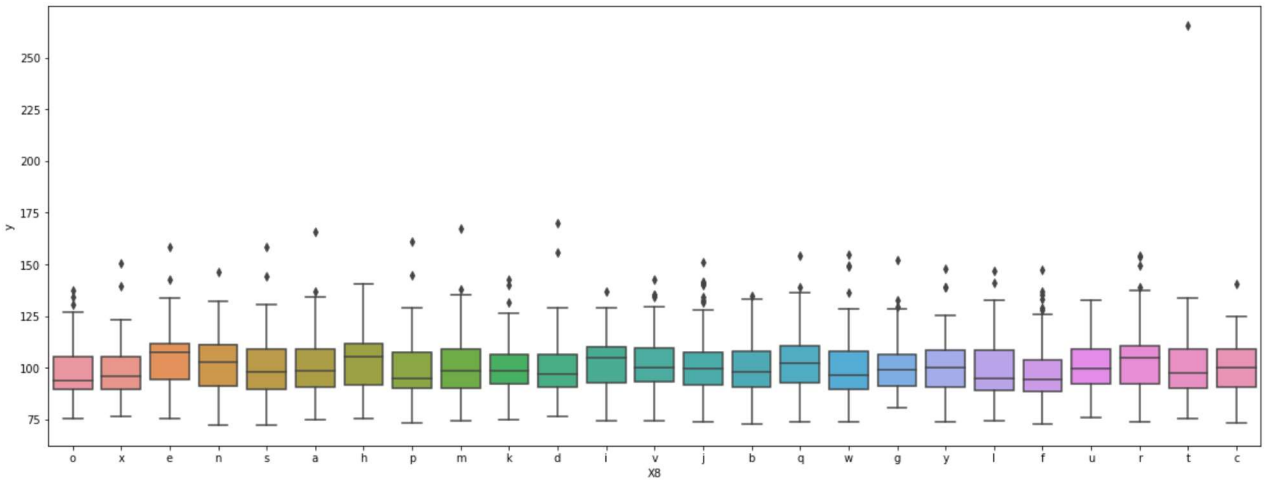

y
X3
a e c f d b g


y
X4
d b c a


y







In [29]:
```
pd.options.display.float_format='{:.2f}'.format
```

If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

In [30]:
```
train_data.var()
```

Out[30]:
```
ID      5941936.12
y           160.77
X10           0.01
X11           0.00
X12           0.07
           ...
X380          0.01
X382          0.01
X383          0.00
X384          0.00
X385          0.00
Length: 370, dtype: float64
```

In [31]:
```
train_data['X383'].var()
```

Out[31]:
```
0.0016607315303888506
```

In [32]:
```
len(np.unique(train_data['X383']))
```

Out[32]:
```
2
```

In [33]:
```
non_zero_var=[]
zero_var=[]
for cols in binary_cols:
    if train_data[cols].var()!=float(0.00):
        non_zero_var.append(cols)
    else:
        zero_var.append(cols)
```

In [34]:
```
len(non_zero_var)
```

Out[34]:
```
356
```

In [35]:
```
zero_var
```

Out[35]:
```
['X11',
 'X93',
 'X107',
 'X233',
 'X235',
 'X268',
 'X289',
 'X290',
 'X293',
 'X297',
 'X330',
 'X347']
```

In [36]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
```

In [44]:
```
X_train=train_data.drop(['ID','y'], axis=1)
X_test=test_data.drop('ID', axis=1)
```

In [45]:
```
Y_train=train_data['y']
```

In [46]:
```
X_train.head()
```

Out[46]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X3

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X3
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0	
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0	
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1	
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0	
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0	

5 rows × 376 columns

Apply label encoder.

In [52]:
```
le=LabelEncoder()
for col in cat_cols:
    X_train[col]=le.fit_transform(X_train[col])
    X_test[col]=le.fit_transform(X_test[col])
```

In [164...
```
X_train.head()
```

Out[164...

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X3
0	32	23	17	0	3	24	9	14	0	0	...	0	0	1	0	0	0	0	
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	0	0	
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	0	1	
3	20	21	34	5	3	27	11	4	0	0	...	0	0	0	0	0	0	0	
4	20	23	34	5	3	12	3	13	0	0	...	0	0	0	0	0	0	0	

5 rows × 376 columns

Perform dimensionality reduction.

In [133...
```
pca=PCA(n_components=2, random_state=42)
pca_train=pca.fit_transform(X_train)
pca_test=pca.transform(X_test)
```

In [134...
```
pca.explained_variance_ratio_
```

Out[134... array([0.38334782, 0.21388033])

Predict your test_df values using xgboost

In [135…
```
import xgboost
xgbr=xgboost.XGBRegressor()
```

In [136…
```
xgbr.fit(pca_train, Y_train)
```

Out[136…
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [137…
```
Y_pred=xgbr.predict(pca_test)
```

In [138…
```
from sklearn.metrics import mean_squared_error, r2_score
mse=mean_squared_error(Y_train,Y_pred)
r2=r2_score(Y_train, Y_pred)
```

In [139…
```
print(f"mse:%s r2:%s"%(mse,r2))
```

```
mse:218.80401049554365 r2:-0.3613266582294923
```

In [140…
```
df=pd.DataFrame({'ID':test_data['ID'], 'ACTUAL_VALUES':Y_train, 'PREDICTED_VALUES':Y_pr
```

In [142…
```
df.to_csv('result.csv')
```

In [ ]:
```

```