In [1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import plotly
import matplotlib.pyplot as plt
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

In [2]:

```python
df = pd.read_csv('C:/Users/User/Desktop/datasciernce/employee_promotion.csv')
```

In [3]:

```python
df.head()
```

Out[3]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_training
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	
1	65141	Operations	region_22	Bachelor's	m	other	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	
4	48945	Technology	region_26	Bachelor's	m	other	

In [4]:

```python
df.shape
```

Out[4]:

```
(54808, 13)
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   employee_id           54808 non-null  int64  
 1   department            54808 non-null  object 
 2   region                54808 non-null  object 
 3   education             52399 non-null  object 
 4   gender                54808 non-null  object 
 5   recruitment_channel   54808 non-null  object 
 6   no_of_trainings       54808 non-null  int64  
 7   age                   54808 non-null  int64  
 8   previous_year_rating  50684 non-null  float64
 9   length_of_service     54808 non-null  int64  
 10  awards_won            54808 non-null  int64  
 11  avg_training_score    52248 non-null  float64
 12  is_promoted           54808 non-null  int64  
dtypes: float64(2), int64(6), object(5)
memory usage: 5.4+ MB
```

In [6]:

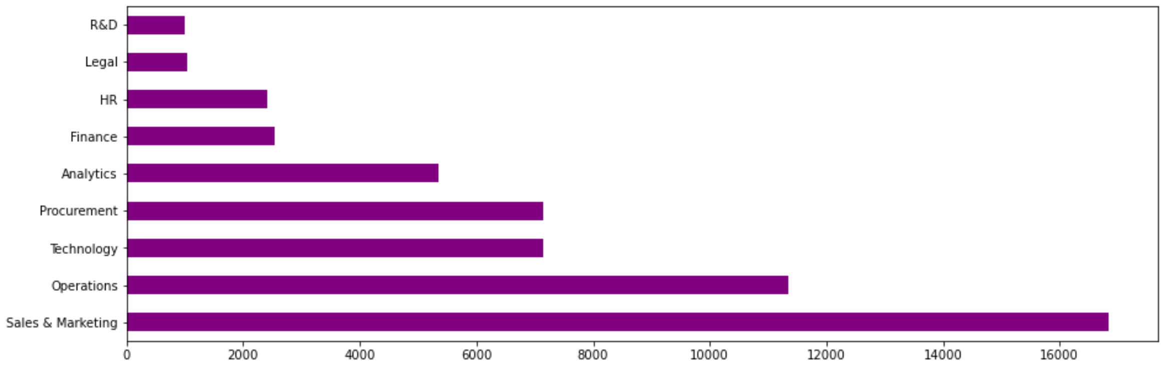
```
df['department'].value_counts()
```

Out[6]:

```
Sales & Marketing    16840
Operations           11348
Technology            7138
Procurement           7138
Analytics             5352
Finance               2536
HR                    2418
Legal                 1039
R&D                    999
Name: department, dtype: int64
```

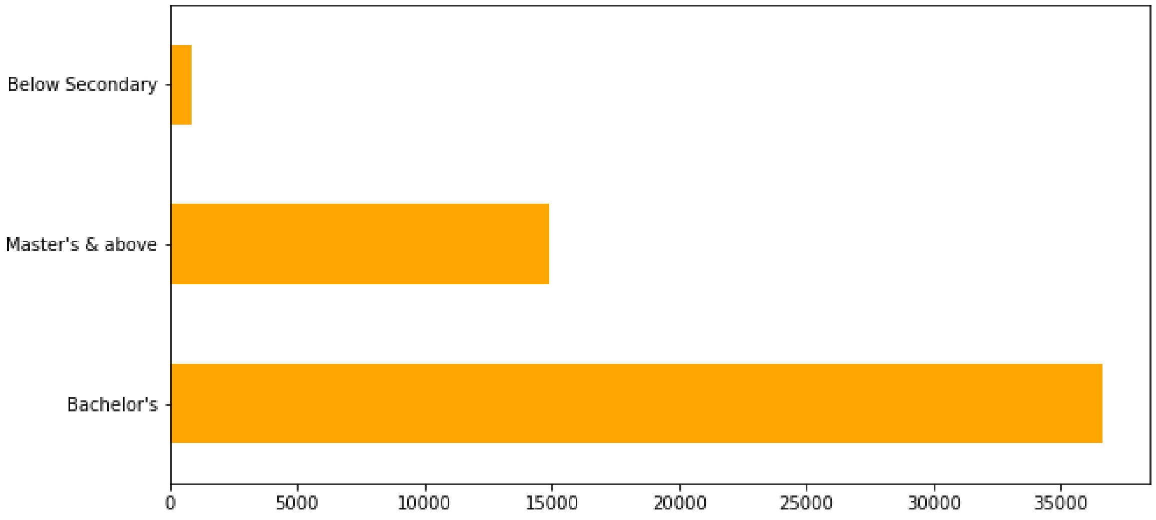
In [7]:

```
plt.figure(figsize=(15,5))
df['department'].value_counts().plot(kind='barh',color='purple')
plt.show()
```



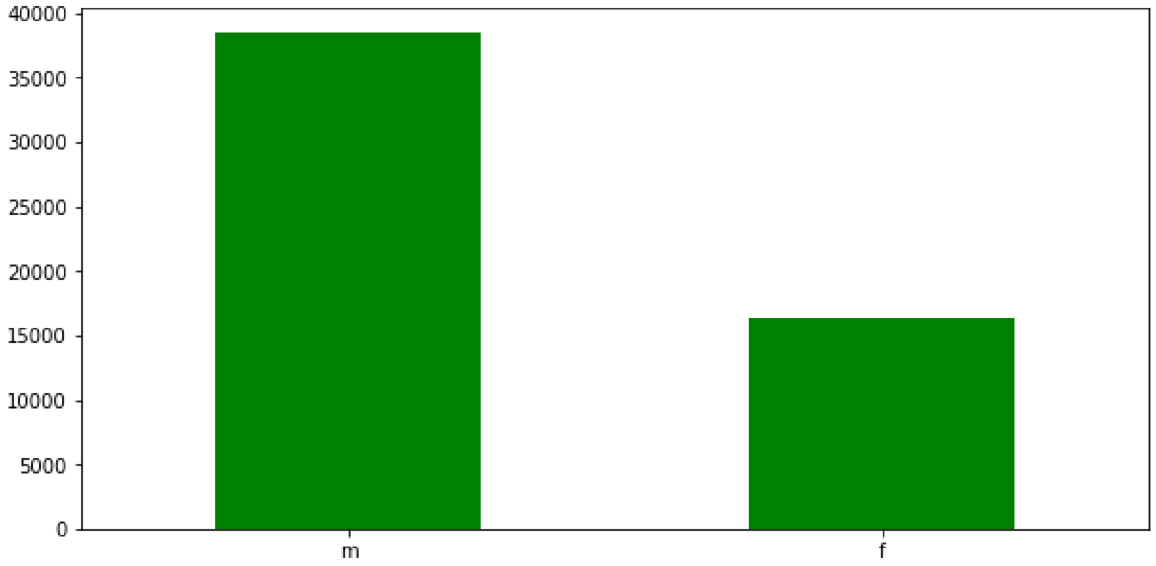
In [8]:

```
plt.figure(figsize=(10,5))
df['education'].value_counts().plot(kind='barh',color='orange')
plt.show()
```



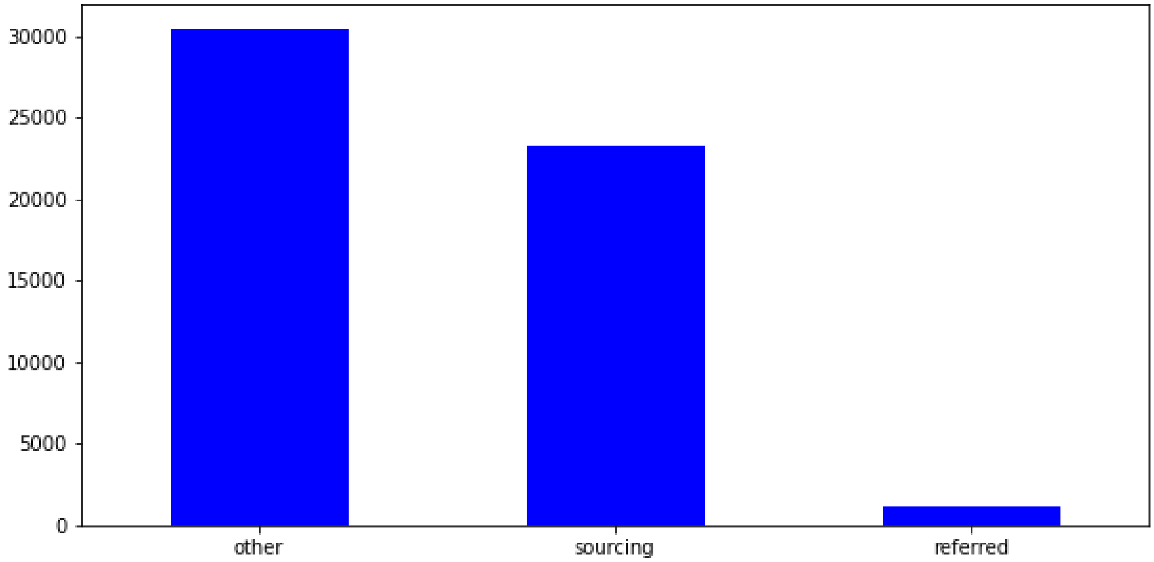
In [9]:

```
plt.figure(figsize=(10,5))
df['gender'].value_counts().plot(kind='bar',color='green')
plt.xticks(rotation=0)
plt.show()
```



In [10]:

```python
plt.figure(figsize=(10,5))
df['recruitment_channel'].value_counts().plot(kind='bar',color='blue')
plt.xticks(rotation=0)
plt.show()
```



Encoding

In [11]:

```python
le = LabelEncoder()
```

In [12]:

```python
df['department']=le.fit_transform(df['department'])
```

In [13]:

```python
df['region']=le.fit_transform(df['region'])
df['education']=le.fit_transform(df['education'])
df['recruitment_channel']=le.fit_transform(df['recruitment_channel'])
df['gender']=le.fit_transform(df['gender'])
```

In [14]:

```
df.head()
```

Out[14]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings
0	65438	7	31	2	0	2	1
1	65141	4	14	0	1	0	1
2	7513	7	10	0	1	2	1
3	2542	7	15	0	1	0	2
4	48945	8	18	0	1	0	1

In [15]:

```
df = df.drop('employee_id',axis=1)
```

In [16]:

```
cols = [ 'previous_year_rating', 'avg_training_score']
for col in cols:
   df[col] = df[col].apply(lambda x: int(x) if x == x else 0)
```

In [17]:

```
df.head()
```

Out[17]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous
0	7	31	2	0	2	1	35	
1	4	14	0	1	0	1	30	
2	7	10	0	1	2	1	34	
3	7	15	0	1	0	2	39	
4	8	18	0	1	0	1	45	

In [18]:

```
df = df.fillna(0)
```

In [19]:

```
df.isnull().any()
```

Out[19]:

```
department               False
region                   False
education                False
gender                   False
recruitment_channel      False
no_of_trainings          False
age                      False
previous_year_rating     False
length_of_service        False
awards_won               False
avg_training_score       False
is_promoted              False
dtype: bool
```

In [20]:

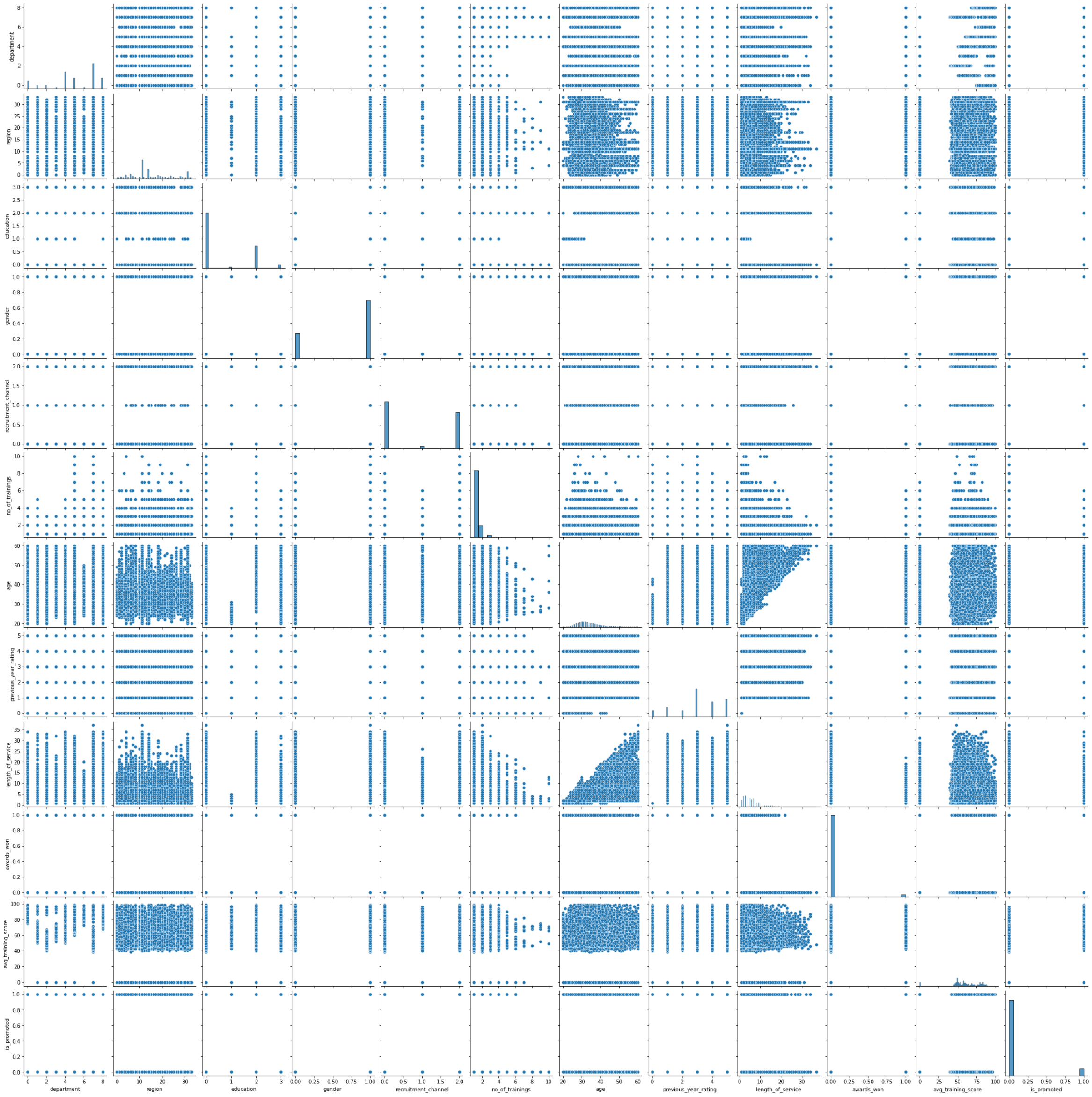
```
df['region'].value_counts()
```

Out[20]:

```
11    12343
14     6428
31     4843
6      2808
4      2648
18     2260
24     1935
28     1703
19     1659
7      1465
20     1318
2      1315
15     1175
21      994
25      945
10      874
12      850
5       827
17      819
8       796
29      766
30      690
23      657
32      655
1       648
0       610
16      508
3       500
33      420
13      411
22      346
27      292
26      269
9        31
Name: region, dtype: int64
```

```
In [21]: sns.pairplot(df)
```

```
Out[21]: <seaborn.axisgrid.PairGrid at 0x23a5d9ce250>
```

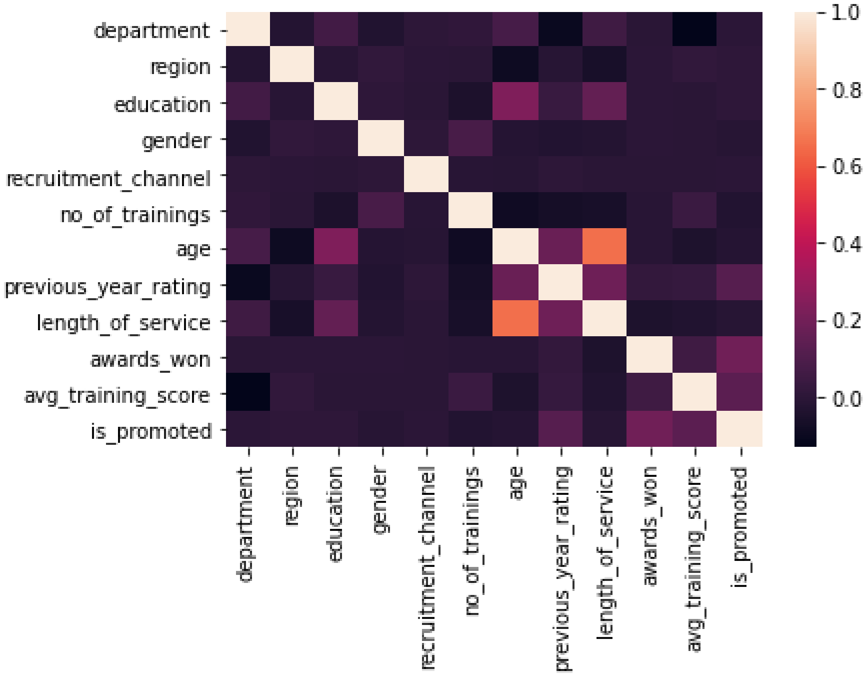


In [22]:

```
tc = df.corr()
sns.heatmap(tc)
```

Out[22]:

```
<AxesSubplot:>
```



In [23]:

```
df.columns
```

Out[23]:

```
Index(['department', 'region', 'education', 'gender', 'recruitment_channe
l',
       'no_of_trainings', 'age', 'previous_year_rating', 'length_of_servic
e',
       'awards_won', 'avg_training_score', 'is_promoted'],
      dtype='object')
```

In [24]:

```
X = df[['department', 'region', 'education', 'gender', 'recruitment_channel',
       'no_of_trainings', 'age', 'previous_year_rating', 'length_of_service',
       'awards_won', 'avg_training_score']]
Y = df[['is_promoted']]
```

In [25]:

```
print(X.shape)
print(Y.shape)
```

```
(54808, 11)
(54808, 1)
```

Spliting the dataset as train and test(test size of 30%)

In [26]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3,random_s
```

In [27]:

```
print(X_train.shape)
print(Y_train.shape)
```

```
(38365, 11)
(38365, 1)
```

Building the classification model

In [28]:

```
from sklearn.linear_model import LogisticRegression
```

In [29]:

```
model_lr = LogisticRegression()
```

In [30]:

```
model_lr.fit(X_train,Y_train)
```

```
C:\xamp\anaconda1\lib\site-packages\sklearn\utils\validation.py:63: DataCon
versionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\xamp\anaconda1\lib\site-packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sci
kit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  n_iter_i = _check_optimize_result(
```

Out[30]:

```
LogisticRegression()
```

In [31]:

```
pred = model_lr.predict(X_test)
```

In [32]:

```
pred
```

Out[32]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [33]:

```
from sklearn import metrics
```

In [34]:

```
print('The accuracy is ',(metrics.accuracy_score(Y_test,pred)*100),'%')
```

```
The accuracy is  91.60737091771574 %
```

In [35]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [36]:

```
model = DecisionTreeClassifier()
```

In [37]:

```
model.fit(X_train,Y_train)
```

Out[37]:

```
DecisionTreeClassifier()
```

In [39]:

```
Pred = model.predict(X_test)
```

In [40]:

```
print('The accuracy is ',(metrics.accuracy_score(Y_test,Pred)*100),'%')
```

```
The accuracy is  87.71513714042449 %
```

In [41]:

```
pip install xgboost
```

```
Collecting xgboostNote: you may need to restart the kernel to use updated p
ackages.

  Using cached xgboost-1.4.2-py3-none-win_amd64.whl (97.8 MB)
Requirement already satisfied: scipy in c:\xamp\anaconda1\lib\site-packages
(from xgboost) (1.6.2)
Requirement already satisfied: numpy in c:\xamp\anaconda1\lib\site-packages
(from xgboost) (1.20.1)
Installing collected packages: xgboost
Successfully installed xgboost-1.4.2
```

In [42]:

```
import xgboost
```

In [43]:

```
from xgboost import XGBClassifier
```

In [44]:

```
model_xg = XGBClassifier()
```

In [45]:

```
model_xg.fit(X_train,Y_train)
```

```
C:\xamp\anaconda1\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: T
he use of label encoder in XGBClassifier is deprecated and will be removed
in a future release. To remove this warning, do the following: 1) Pass opti
on use_label_encoder=False when constructing XGBClassifier object; and 2) E
ncode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_
class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\xamp\anaconda1\lib\site-packages\sklearn\utils\validation.py:63: DataCon
versionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)

[11:33:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluatio
n metric used with the objective 'binary:logistic' was changed from 'error'
to 'logloss'. Explicitly set eval_metric if you'd like to restore the old b
ehavior.
```

Out[45]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', 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=4, num_parallel_tree=1, random_state
=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [46]:

```
predicted = model_xg.predict(X_test)
```

In [47]:

```
print('The accuracy is ',(metrics.accuracy_score(Y_test,predicted)*100),'%')
```

```
The accuracy is  94.16164933406313 %
```

In [ ]:

```

```