In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('digit_data.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pix
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

In [4]:

```
df.tail()
```

Out[4]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774
41995	0	0	0	0	0	0	0	0	0	0	...	0
41996	1	0	0	0	0	0	0	0	0	0	...	0
41997	7	0	0	0	0	0	0	0	0	0	...	0
41998	6	0	0	0	0	0	0	0	0	0	...	0
41999	9	0	0	0	0	0	0	0	0	0	...	0

5 rows × 785 columns

In [5]:

```
df.shape
```

Out[5]:

```
(42000, 785)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel
5',
       'pixel6', 'pixel7', 'pixel8',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel7
79',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
label       0
pixel0      0
pixel1      0
pixel2      0
pixel3      0
           ..
pixel779    0
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 785, dtype: int64
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42000 entries, 0 to 41999
Columns: 785 entries, label to pixel783
dtypes: int64(785)
memory usage: 251.5 MB
```

In [10]:

```
df.describe()
```

Out[10]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7
count	42000.000000	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0
mean	4.456643	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
std	2.887730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

8 rows × 785 columns

In [11]:

```
df.nunique()
```

Out[11]:

```
label       10
pixel0       1
pixel1       1
pixel2       1
pixel3       1
           ..
pixel779     3
pixel780     1
pixel781     1
pixel782     1
pixel783     1
Length: 785, dtype: int64
```

In [12]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:

```
import numpy as np
```

In [14]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [15]:

```
df['label'].unique()
```

Out[15]:

```
array([1, 0, 4, 7, 3, 5, 8, 9, 2, 6], dtype=int64)
```

In [16]:

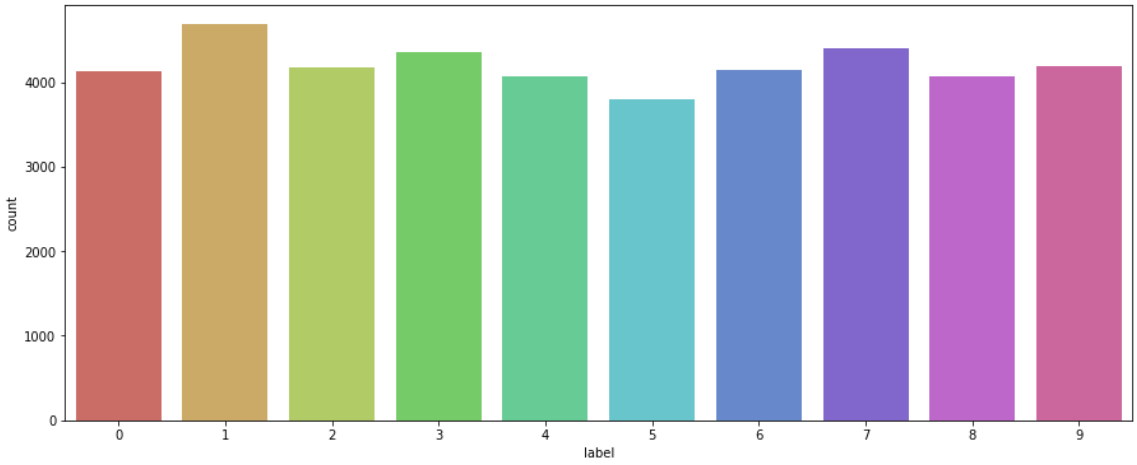
```
df['label'].value_counts()
```

Out[16]:

```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

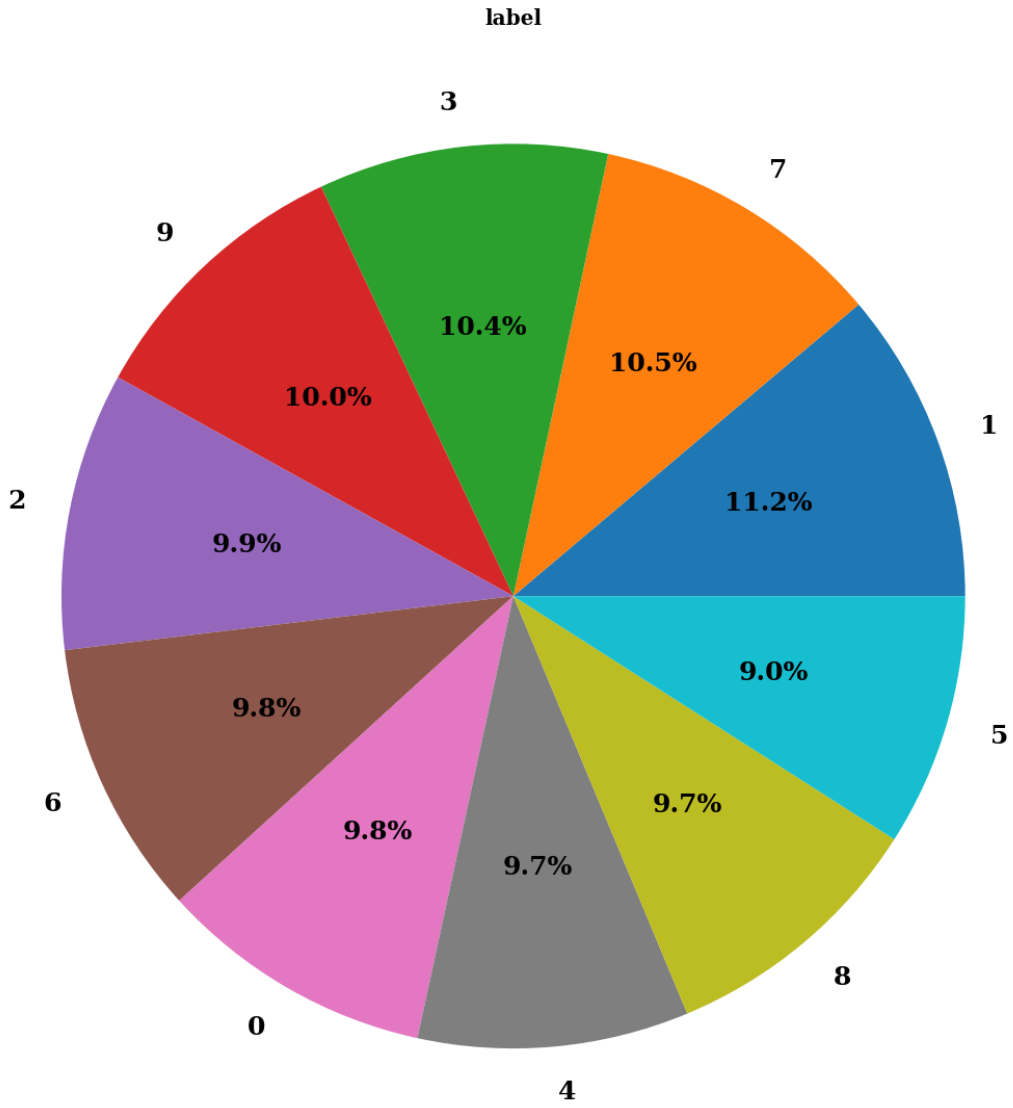
In [17]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['label'], data = df, palette = 'hls')
plt.show()
```



In [18]:

```
plt.figure(figsize=(30,20))
plt.pie(df['label'].value_counts(), labels=df['label'].value_counts().index, autopct='%1
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('label', size=20, **hfont)
plt.show()
```



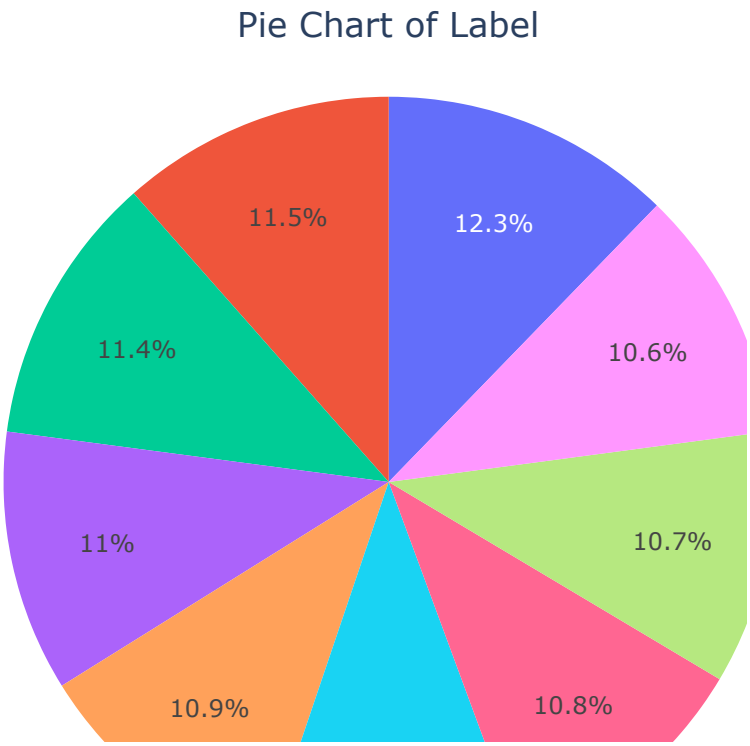
In [20]:

```
import plotly.express as px

fig = px.bar(df, x="label", y= df.index, color = 'label')
fig.show()
```

In [21]:

```
value_counts = df['label'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Label',
    title_x=0.5
)
fig.show()
```



In [22]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [23]:

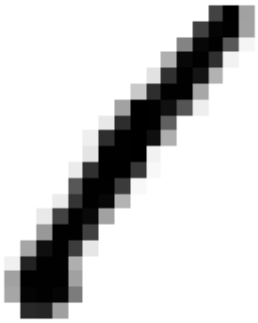
```
X = df.drop(["label"],axis=1)
y = df['label']
```

In [24]:

```
first_image = X.iloc[0]
first_image = first_image.to_numpy().reshape(28,28)
```

In [26]:

```
plt.imshow(first_image, cmap='binary')
plt.axis("off")
plt.show()
```



In [27]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state =
```

In [28]:

```
lrc = LogisticRegression(multi_class = "multinomial")
lrc.fit(X_train,y_train)
```

Out[28]:

```
LogisticRegression
LogisticRegression(multi_class='multinomial')
```

In [29]:

```
some_digit = X_test.iloc[[0]]
some_digit_pred = lrc.predict(some_digit)
some_digit_pred
```

Out[29]:

```
array([8], dtype=int64)
```

In [30]:

```
y_test.iloc[[0]]
```

Out[30]:

```
5457    8
Name: label, dtype: int64
```

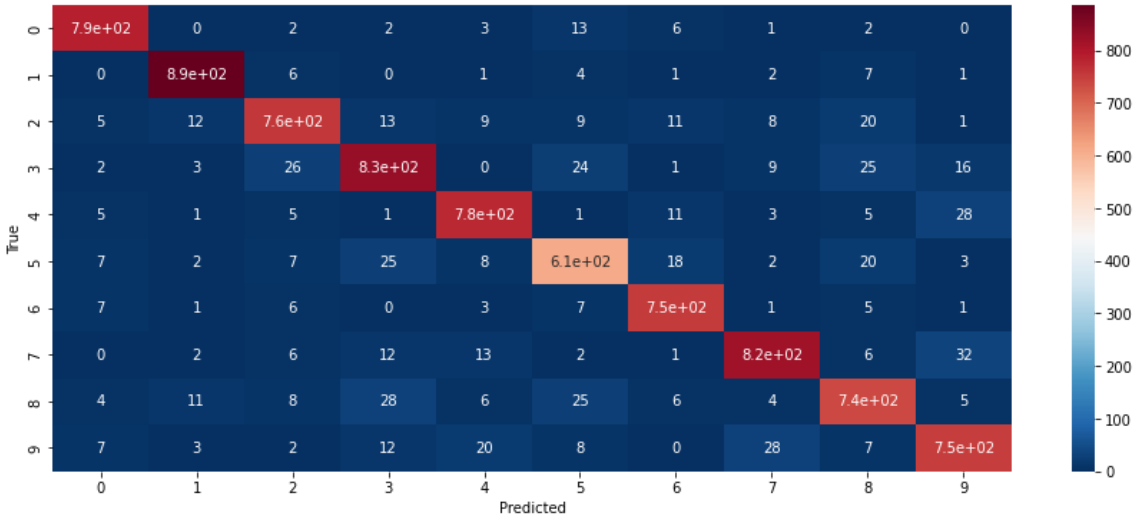
In [31]:

```
y_pred = lrc.predict(X_test)
```

In [36]:

```
print(accuracy_score(y_test, y_pred))
confusion = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(15,6))
sns.heatmap(confusion, annot=True, cmap="RdBu_r")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
0.9183333333333333
```



In [37]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       816
           1       0.96      0.98      0.97       909
           2       0.92      0.90      0.91       846
           3       0.90      0.89      0.89       937
           4       0.93      0.93      0.93       839
           5       0.87      0.87      0.87       702
           6       0.93      0.96      0.95       785
           7       0.93      0.92      0.93       893
           8       0.88      0.88      0.88       835
           9       0.90      0.90      0.90       838

    accuracy                           0.92      8400
   macro avg       0.92      0.92      0.92      8400
weighted avg       0.92      0.92      0.92      8400
```

In [38]:

```
dtc = DecisionTreeClassifier(random_state = 42)
dtc.fit(X_train,y_train)
```

Out[38]:

```
DecisionTreeClassifier(random_state=42)
```

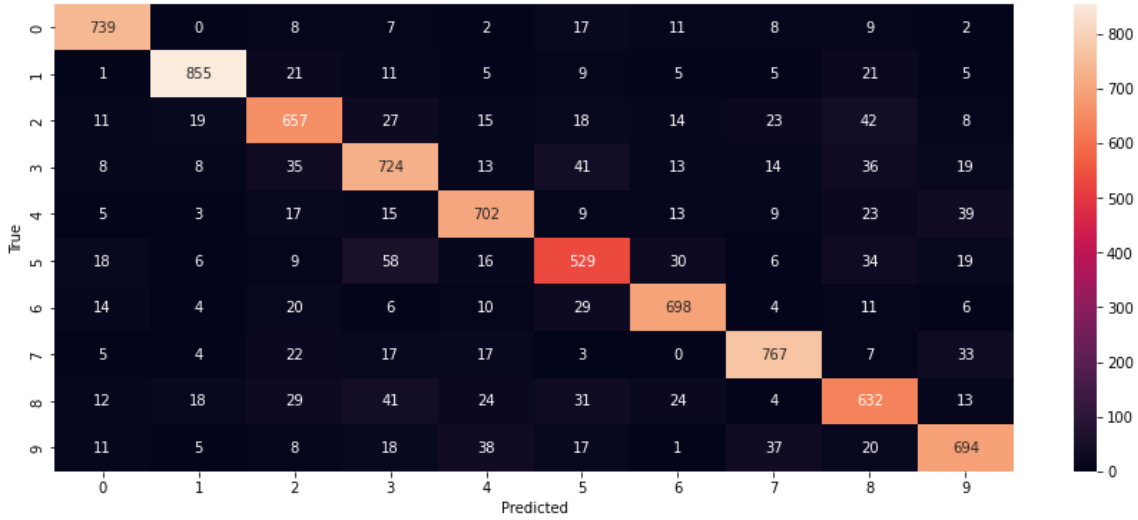
In [39]:

```
dtc_pred = dtc.predict(X_test)
```

In [40]:

```
print(accuracy_score(y_test,dtc_pred))
confusion = confusion_matrix(dtc_pred, y_pred)
plt.figure(figsize=(15,6))
sns.heatmap(confusion, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
0.8525
```



In [41]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,dtc_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       816
           1       0.91      0.94      0.93       909
           2       0.82      0.81      0.81       846
           3       0.83      0.81      0.82       937
           4       0.85      0.85      0.85       839
           5       0.77      0.79      0.78       702
           6       0.87      0.89      0.88       785
           7       0.90      0.88      0.89       893
           8       0.79      0.78      0.78       835
           9       0.84      0.85      0.84       838

    accuracy                           0.85      8400
   macro avg       0.85      0.85      0.85      8400
weighted avg       0.85      0.85      0.85      8400
```

In [42]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state = 42)
rfc.fit(X_train,y_train)
```

Out[42]:

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

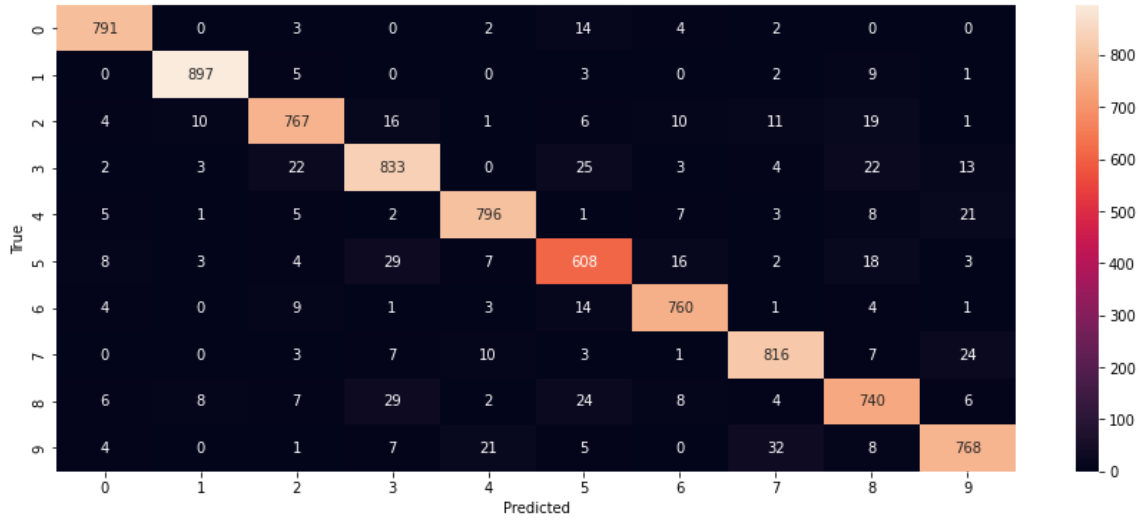
In [43]:

```
rfc_pred = rfc.predict(X_test)
```

In [44]:

```
print(accuracy_score(y_test,dtc_pred))
confusion = confusion_matrix(rfc_pred, y_pred)
plt.figure(figsize=(15,6))
sns.heatmap(confusion, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
0.8525
```



In [45]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       816
           1       0.98      0.99      0.99       909
           2       0.96      0.96      0.96       846
           3       0.96      0.95      0.96       937
           4       0.96      0.97      0.96       839
           5       0.96      0.96      0.96       702
           6       0.96      0.98      0.97       785
           7       0.97      0.95      0.96       893
           8       0.95      0.95      0.95       835
           9       0.93      0.94      0.94       838

    accuracy                           0.96      8400
   macro avg       0.96      0.96      0.96      8400
weighted avg       0.96      0.96      0.96      8400
```

In [ ]:

```

```