CREDIT CARD FRAUD DETECTION

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Data collection

In [2]:
```
#loading data
card_data=pd.read_csv("/home/user/Downloads/card_transdata.csv")
card_data
```

Out[2]:

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_price	repeat_retailer	us
0	57.877857	0.311140	1.945940	1.0	
1	10.829943	0.175592	1.294219	1.0	
2	5.091079	0.805153	0.427715	1.0	
3	2.247564	5.600044	0.362663	1.0	
4	44.190936	0.566486	2.222767	1.0	
...	...	...	...	...	
999995	2.207101	0.112651	1.626798	1.0	
999996	19.872726	2.683904	2.778303	1.0	
999997	2.914857	1.472687	0.218075	1.0	
999998	4.258729	0.242023	0.475822	1.0	
999999	58.108125	0.318110	0.386920	1.0	

1000000 rows × 8 columns

Data preparation

In [3]:
```
df=card_data.head(10000)
df
```

Out[3]:

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_price	repeat_retailer	used
0	57.877857	0.311140	1.945940	1.0	
1	10.829943	0.175592	1.294219	1.0	
2	5.091079	0.805153	0.427715	1.0	
3	2.247564	5.600044	0.362663	1.0	
4	44.190936	0.566486	2.222767	1.0	
...	...	...	...	...	
9995	4.225100	2.678220	0.556858	1.0	

9996	3.614858	0.431593	0.061778	1.0
9997	10.131863	10.262508	2.818090	1.0
9998	16.306236	0.014054	1.904495	1.0
9999	1.292596	0.415847	1.332285	0.0

10000 rows × 8 columns

In [4]:
```
#dimension
df.shape
```

Out[4]:
```
(10000, 8)
```

In [5]:
```
#statistical values
df.describe()
```

Out[5]:

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_price	repeat_retailer	u
count	10000.000000	10000.000000	10000.000000	10000.000000	100
mean	26.374744	4.807107	1.824164	0.879000	
std	58.811106	22.260124	2.762438	0.326144	
min	0.049270	0.000930	0.011373	0.000000	
25%	3.808606	0.305219	0.489052	1.000000	
50%	10.026888	1.006459	1.005754	1.000000	
75%	25.866217	3.313696	2.091252	1.000000	
max	2033.498174	990.070315	65.150879	1.000000	

In [6]:
```
#column names
df.columns
```

Out[6]:
```
Index(['distance_from_home', 'distance_from_last_transaction',
       'ratio_to_median_purchase_price', 'repeat_retailer', 'used_chip',
       'used_pin_number', 'online_order', 'fraud'],
      dtype='object')
```

In [7]:
```
#datatype of each columns
df.dtypes
```

Out[7]:
```
distance_from_home                float64
distance_from_last_transaction    float64
ratio_to_median_purchase_price    float64
repeat_retailer                   float64
used_chip                         float64
used_pin_number                   float64
online_order                      float64
fraud                             float64
dtype: object
```

In [8]:
```
#checking null values
df.isnull().sum()
```

Out[8]:
```
distance_from_home                0
distance_from_last_transaction    0
ratio_to_median_purchase_price    0
```

```
repeat_retailer                 0
used_chip                       0
used_pin_number                 0
online_order                    0
fraud                           0
dtype: int64
```

Data visualization

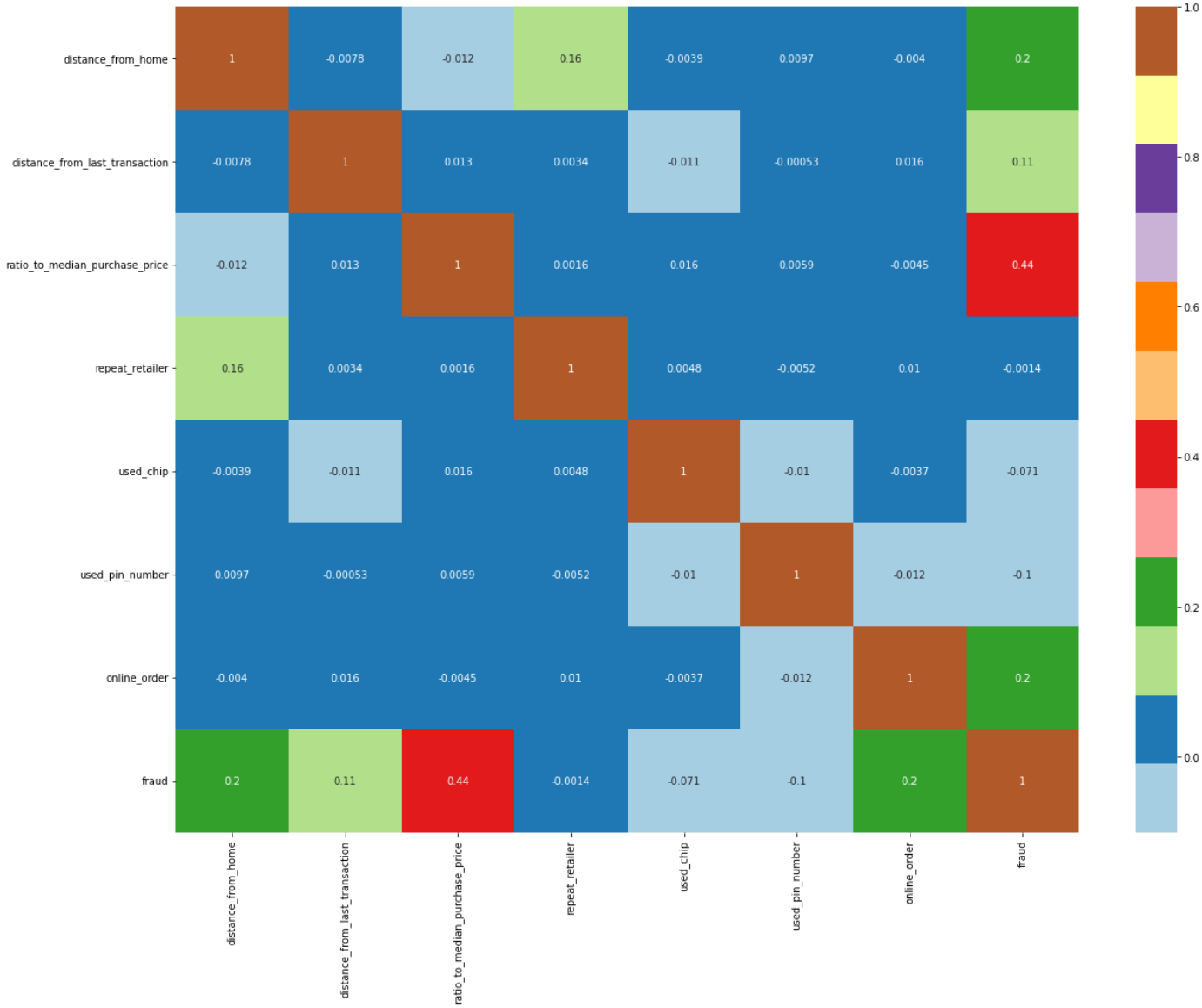
In [9]:
```
#correlation
df.corr()
```

Out[9]:

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_
distance_from_home	1.000000	-0.007818	-0.01
distance_from_last_transaction	-0.007818	1.000000	0.01
ratio_to_median_purchase_price	-0.011728	0.013426	1.00
repeat_retailer	0.159345	0.003432	0.00
used_chip	-0.003907	-0.010870	0.01
used_pin_number	0.009682	-0.000533	0.00
online_order	-0.004010	0.016498	-0.00
fraud	0.195015	0.109744	0.43

In [10]:
```
#heatmap
plt.figure(figsize=(20,15))
sns.heatmap(df.corr(),annot=True,cmap="Paired")
```

Out[10]:
```
<AxesSubplot:>
```

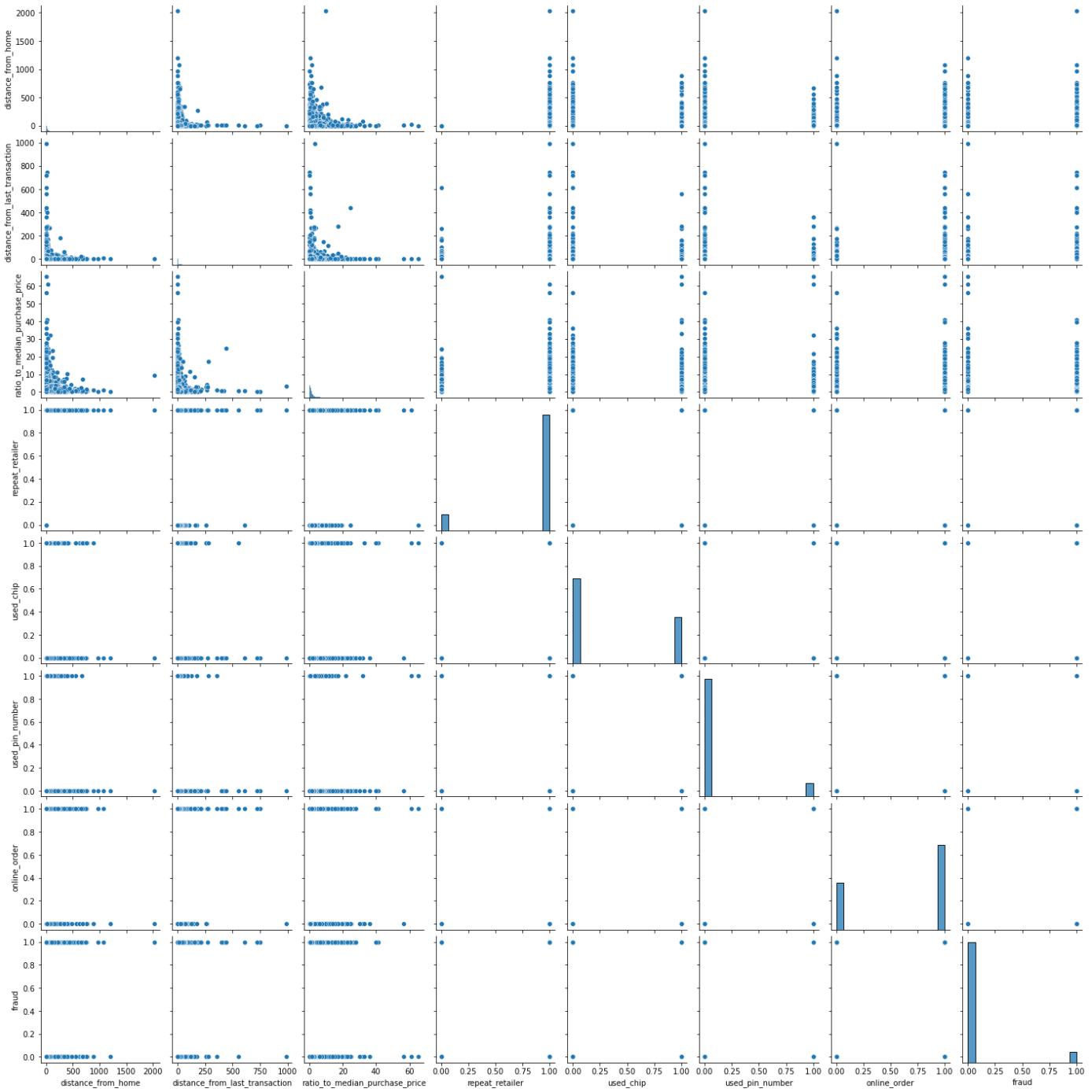


In [11]:

```
#pairplot
sns.pairplot(data=df)
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x7fbcb720aac0>
```



In [12]:

```
#fraud count
f_count=df["fraud"].value_counts()
f_count
```

Out[12]:

```
0.0    9159
1.0     841
Name: fraud, dtype: int64
```

In [13]:

```
#count plot
plt.figure(figsize=(8,8))
sns.countplot("fraud",data=df,palette="Set1")
plt.title("Fraud Count",size="20",weight="bold")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")
```

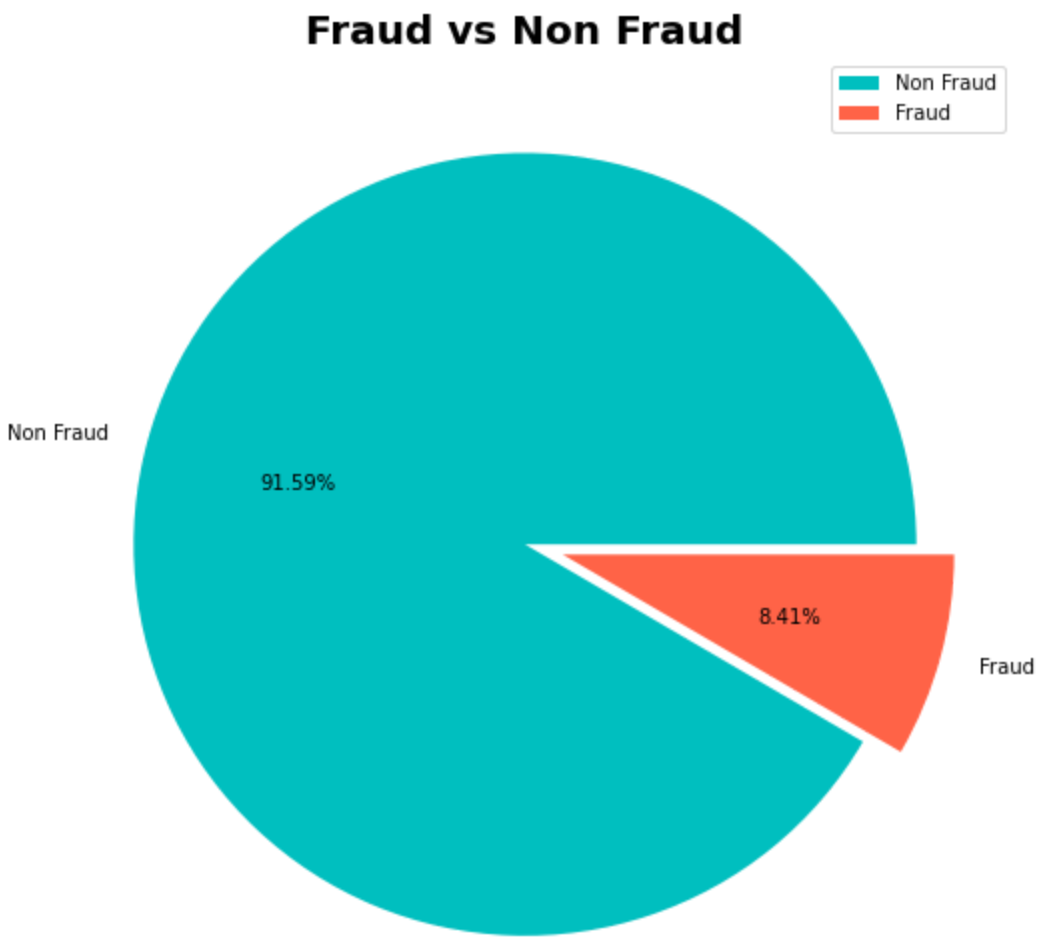
```
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
```

```
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:
```
Text(0, 0.5, 'Count')
```

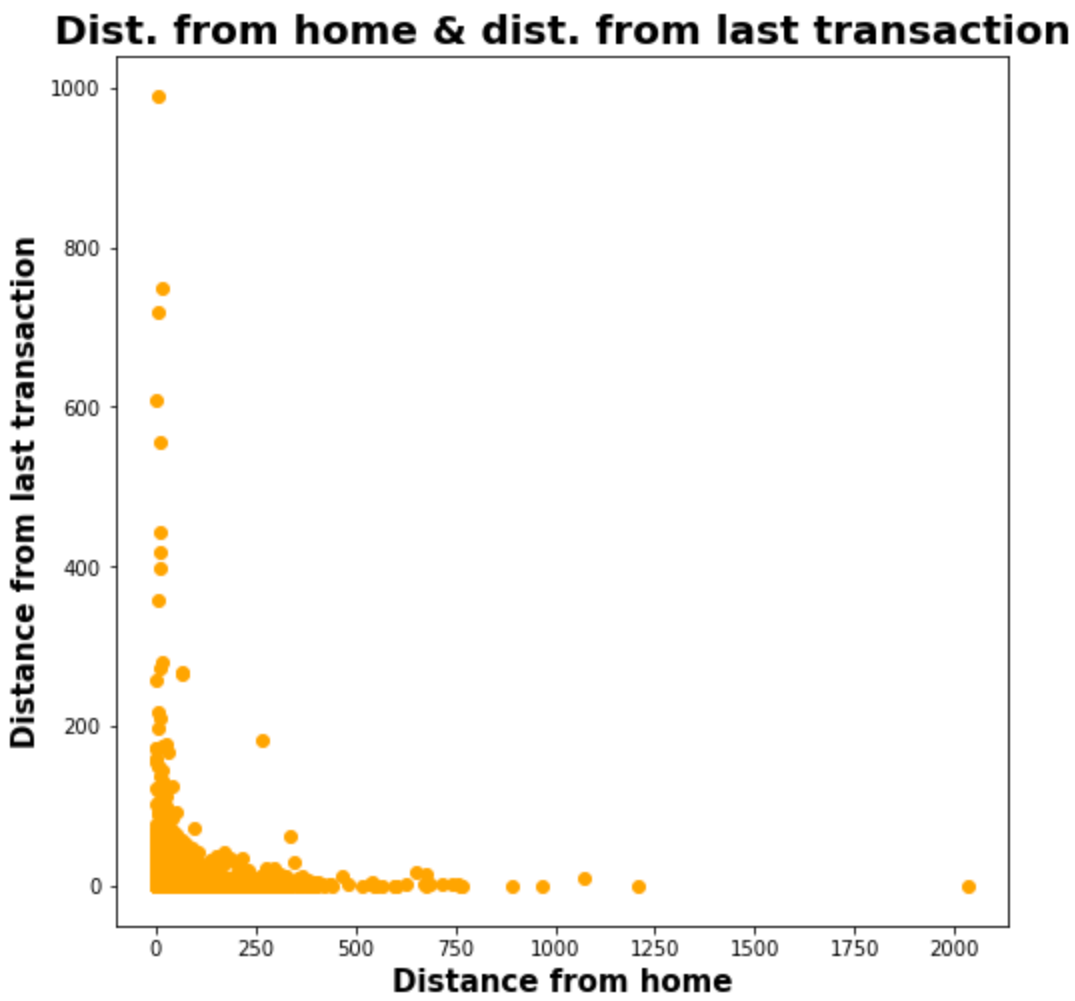
In [14]:
```
#pie plot
plt.figure(figsize=(9,9))
plt.pie(f_count,labels=["Non Fraud","Fraud"],autopct="%.2f%%",explode=(0,0.1),colors=("
plt.title("Fraud vs Non Fraud",size="20",weight="bold")
plt.legend(loc="upper right")
```

Out[14]:
```
<matplotlib.legend.Legend at 0x7fbcae6d77f0>
```



In [15]:
```
#scatter plot
plt.figure(figsize=(8,8))
plt.scatter(df["distance_from_home"],df["distance_from_last_transaction"],color="orange"
plt.title("Dist. from home & dist. from last transaction",size="20",weight="bold")
plt.xlabel("Distance from home",size="15",weight="bold")
plt.ylabel("Distance from last transaction",size="15",weight="bold")
```

Out[15]:
```
Text(0, 0.5, 'Distance from last transaction')
```



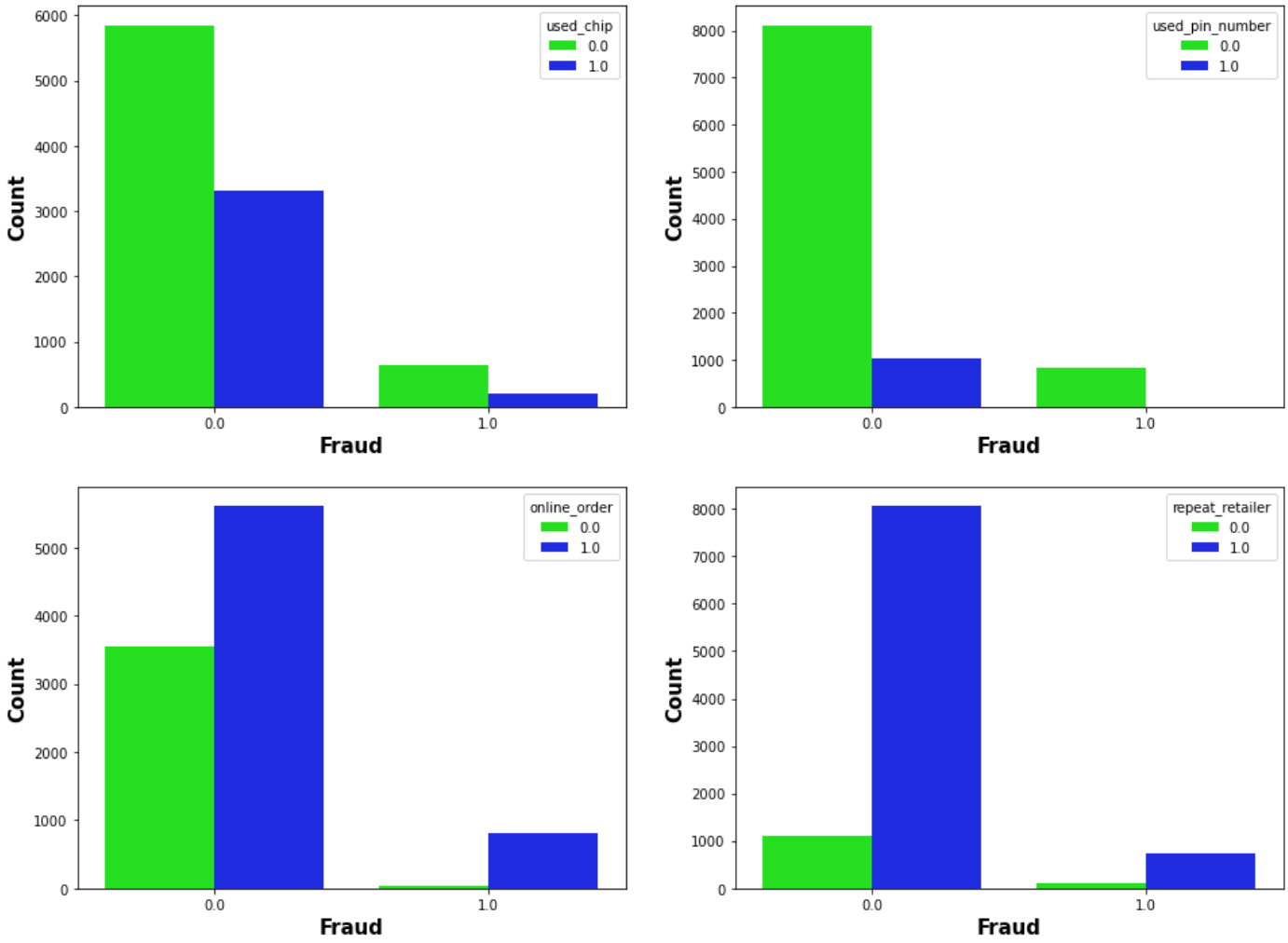
In [16]:

```
#countplot
plt.figure(figsize=(16,12))
plt.subplot(2,2,1)
sns.countplot(df["fraud"],hue=df["used_chip"],palette="hsv")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,2,2)
sns.countplot(df["fraud"],hue=df["used_pin_number"],palette="hsv")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,2,3)
sns.countplot(df["fraud"],hue=df["online_order"],palette="hsv")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,2,4)
sns.countplot(df["fraud"],hue=df["repeat_retailer"],palette="hsv")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")
```

```
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

```
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]: Text(0, 0.5, 'Count')



In [17]:

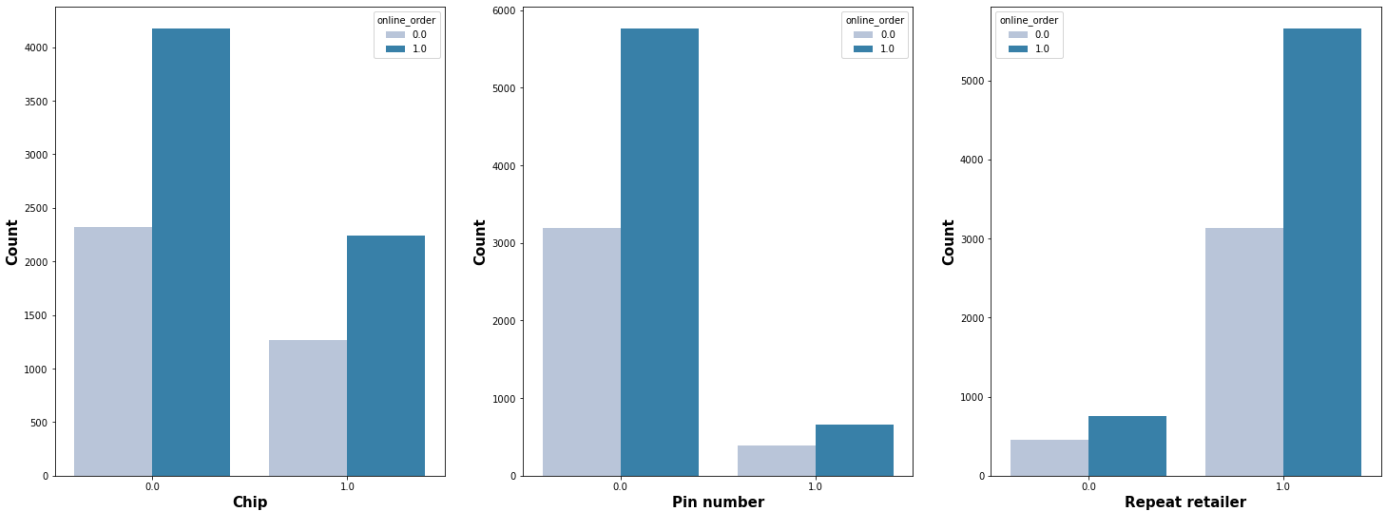
```
#countplot
plt.figure(figsize=(25,20))
plt.subplot(2,3,1)
sns.countplot(df["used_chip"],hue=df["online_order"],palette="PuBu")
plt.xlabel("Chip",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,3,2)
sns.countplot(df["used_pin_number"],hue=df["online_order"],palette="PuBu")
plt.xlabel("Pin number",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,3,3)
sns.countplot(df["repeat_retailer"],hue=df["online_order"],palette="PuBu")
plt.xlabel("Repeat retailer",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,3,4)
sns.countplot(df["used_chip"],hue=df["repeat_retailer"],palette="PuBu")
plt.xlabel("Chip",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")
```

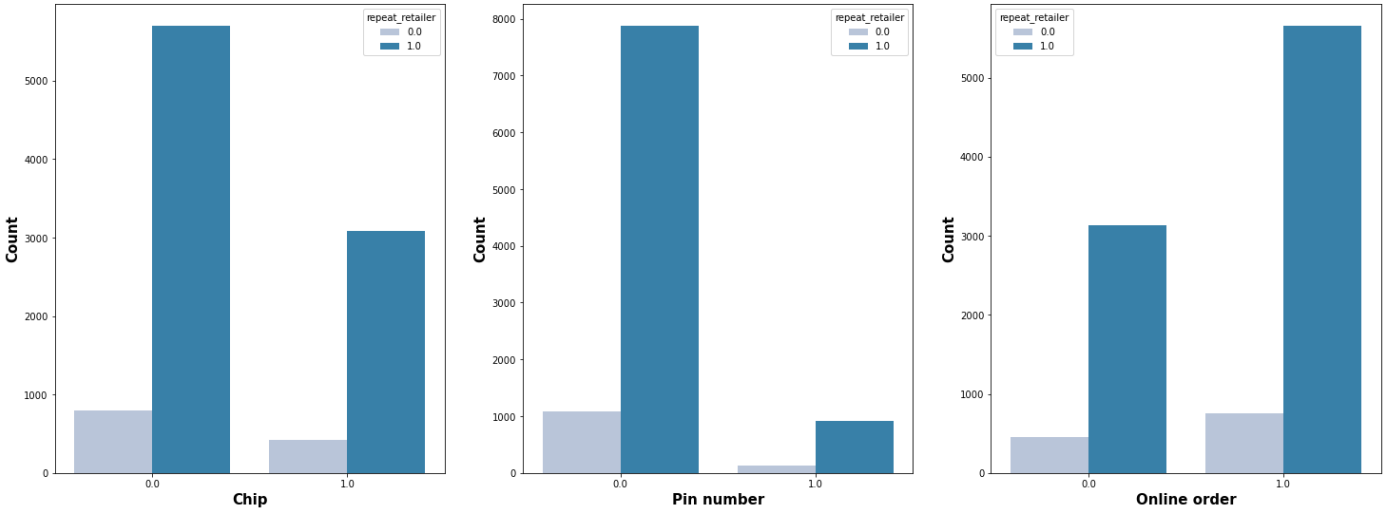
```python
plt.subplot(2,3,5)
sns.countplot(df["used_pin_number"],hue=df["repeat_retailer"],palette="PuBu")
plt.xlabel("Pin number",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")

plt.subplot(2,3,6)
sns.countplot(df["online_order"],hue=df["repeat_retailer"],palette="PuBu")
plt.xlabel("Online order",size="15",weight="bold")
plt.ylabel("Count",size="15",weight="bold")
```

```
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
/home/user/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:

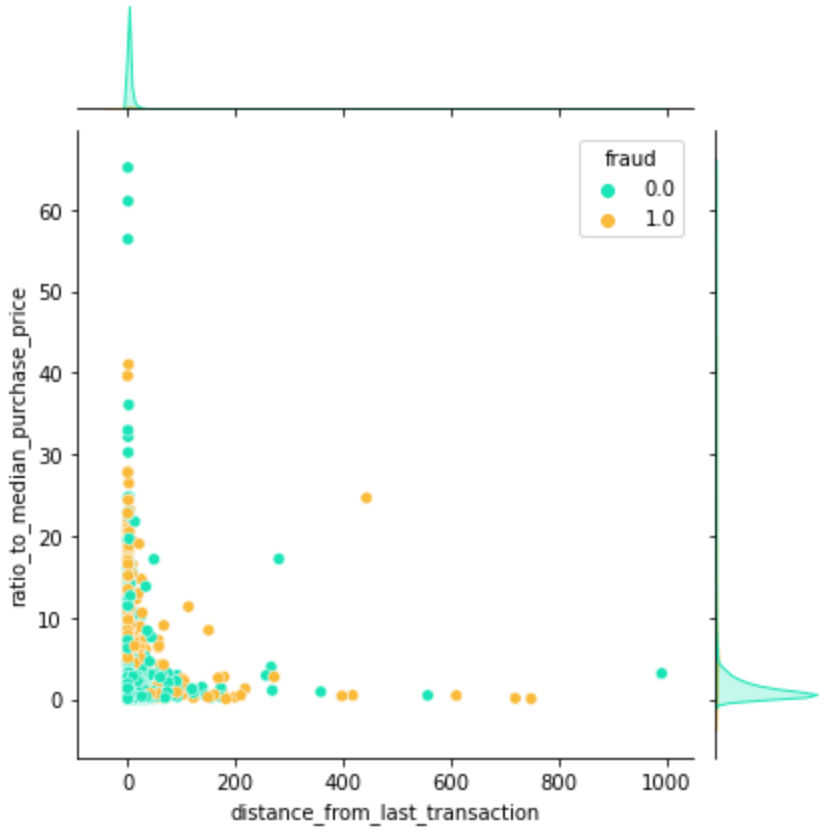
```
Text(0, 0.5, 'Count')
```





In [18]:
```
#jointplot
sns.jointplot(x="distance_from_last_transaction",y="ratio_to_median_purchase_price",data
```

Out[18]:
```
<seaborn.axisgrid.JointGrid at 0x7fbcac5af2e0>
```

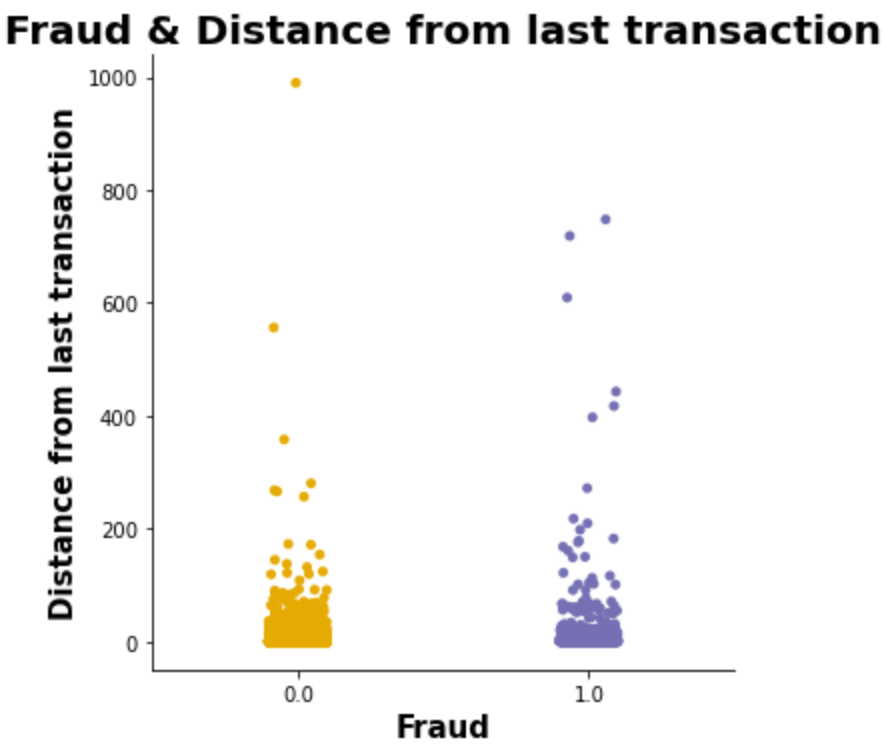


In [19]:

```
#catplot
sns.catplot(x="fraud",y="distance_from_home",data=df,palette="Dark2_r")
plt.title("Fraud & Distance from home",size="20",weight="bold")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Distance from home",size="15",weight="bold")

sns.catplot(x="fraud",y="distance_from_last_transaction",data=df,palette="Dark2_r")
plt.title("Fraud & Distance from last transaction",size="20",weight="bold")
plt.xlabel("Fraud",size="15",weight="bold")
plt.ylabel("Distance from last transaction",size="15",weight="bold")
```

Out[19]:

```
Text(-2.700000000000003, 0.5, 'Distance from last transaction')
```

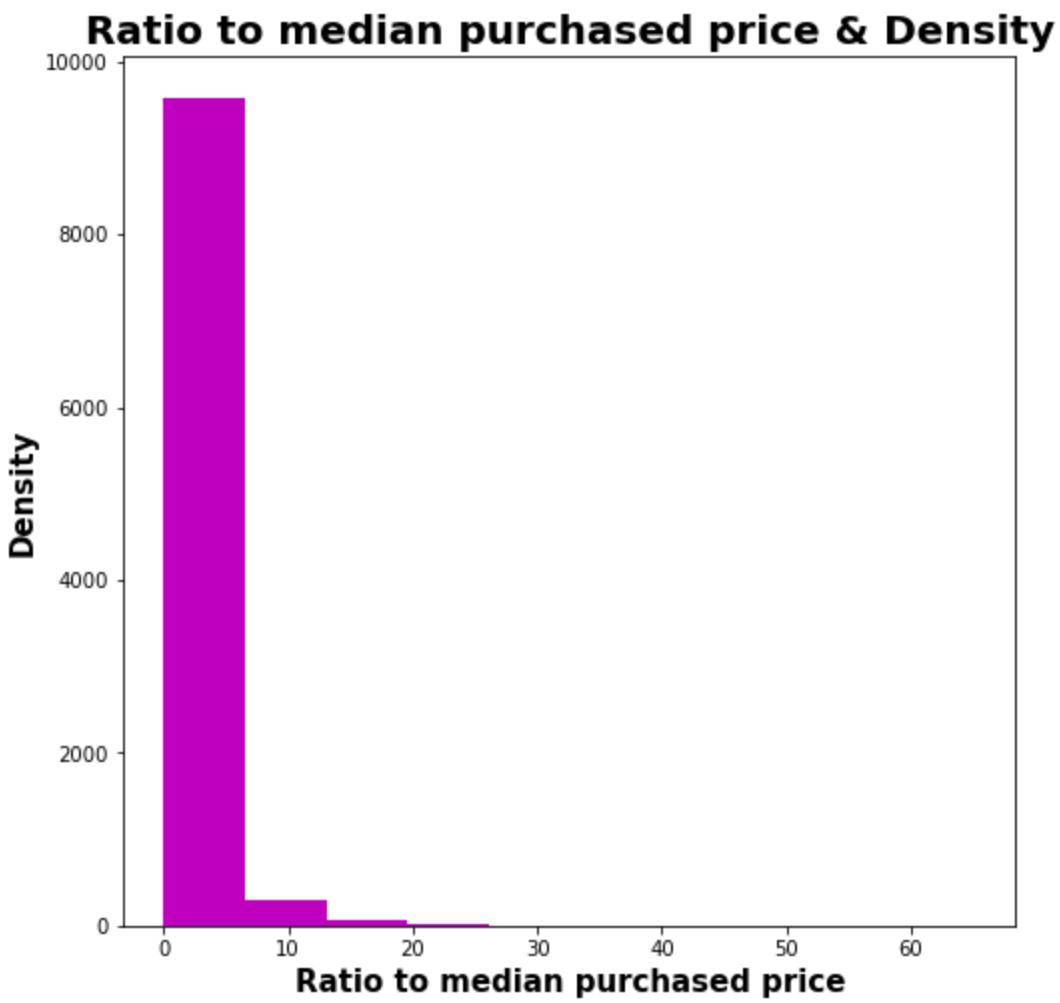


In [20]:

```
#histogram
plt.figure(figsize=(8,8))
plt.hist(df["ratio_to_median_purchase_price"],color="m")
plt.title("Ratio to median purchased price & Density",size="20",weight="bold")
plt.xlabel("Ratio to median purchased price",size="15",weight="bold")
plt.ylabel("Density",size="15",weight="bold")
```

Out[20]:

```
Text(0, 0.5, 'Density')
```



Data modeling

In [21]:
```
#extracting input & output variables
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [22]:
```
#splittings datas into train & test set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=1)
```

In [23]:
```
#standardisation
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In [24]:
```
#implementing Random forest algorithm
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100,criterion="entropy")
rf.fit(x_train,y_train)
```

Out[24]:
```
RandomForestClassifier(criterion='entropy')
```

In [25]:
```
# making prediction
y_pred=rf.predict(x_test)
```

```
y_pred
```

Out[25]:
```
array([0., 1., 0., ..., 0., 0., 1.])
```

In [26]:
```
#comparison between actual & predicted value
act_pred=pd.DataFrame({"Actual value":y_test,"Predicted value":y_pred})
act_pred
```

Out[26]:

	Actual value	Predicted value
0	0.0	0.0
1	1.0	1.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
...	...	...
1995	0.0	0.0
1996	0.0	0.0
1997	0.0	0.0
1998	0.0	0.0
1999	1.0	1.0

2000 rows × 2 columns

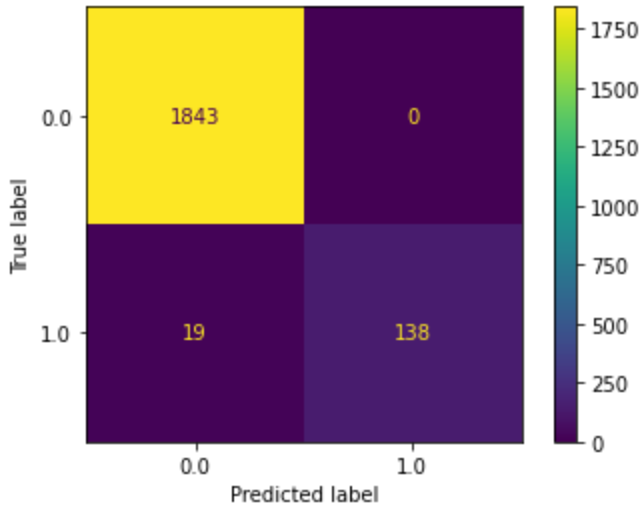
Model evaluation

In [27]:
```
#confustion matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

```
[[1843    0]
 [  19  138]]
```

In [28]:
```
#displaying confusion matrix
from sklearn.metrics import ConfusionMatrixDisplay
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fbcb2697880>
```



In [29]:

```
#classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      1843
         1.0       1.00      0.88      0.94       157

    accuracy                           0.99      2000
   macro avg       0.99      0.94      0.97      2000
weighted avg       0.99      0.99      0.99      2000
```

In [30]:

```
#accuracy
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
0.9905
```

In [ ]: