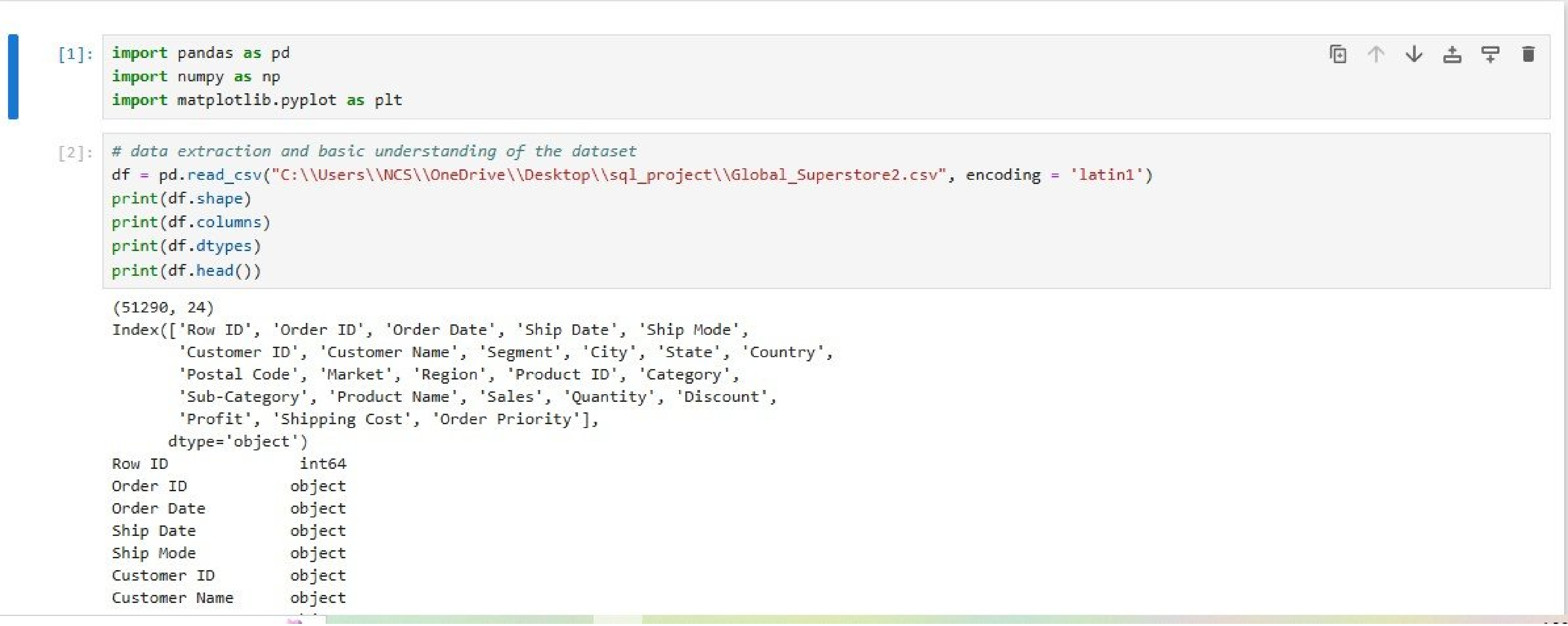


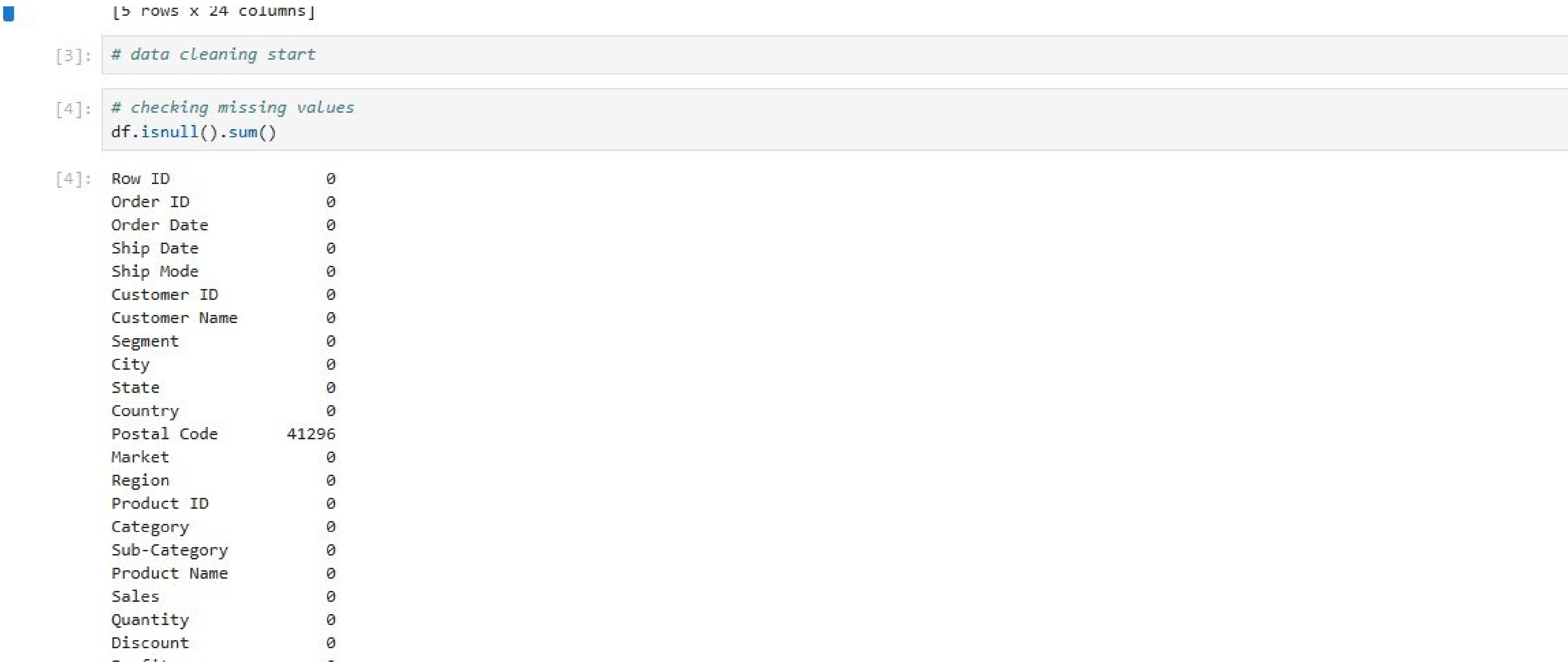
[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:

```
# data extraction and basic understanding of the dataset
df = pd.read_csv("C:\\Users\\NCS\\OneDrive\\Desktop\\sql_project\\Global_Superstore2.csv", encoding = 'latin1')
print(df.shape)
print(df.columns)
print(df.dtypes)
print(df.head())
```

```
(51290, 24)
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
       'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
       'Profit', 'Shipping Cost', 'Order Priority'],
      dtype='object')
Row ID              int64
Order ID           object
Order Date         object
Ship Date          object
Ship Mode          object
Customer ID        object
Customer Name      object
```

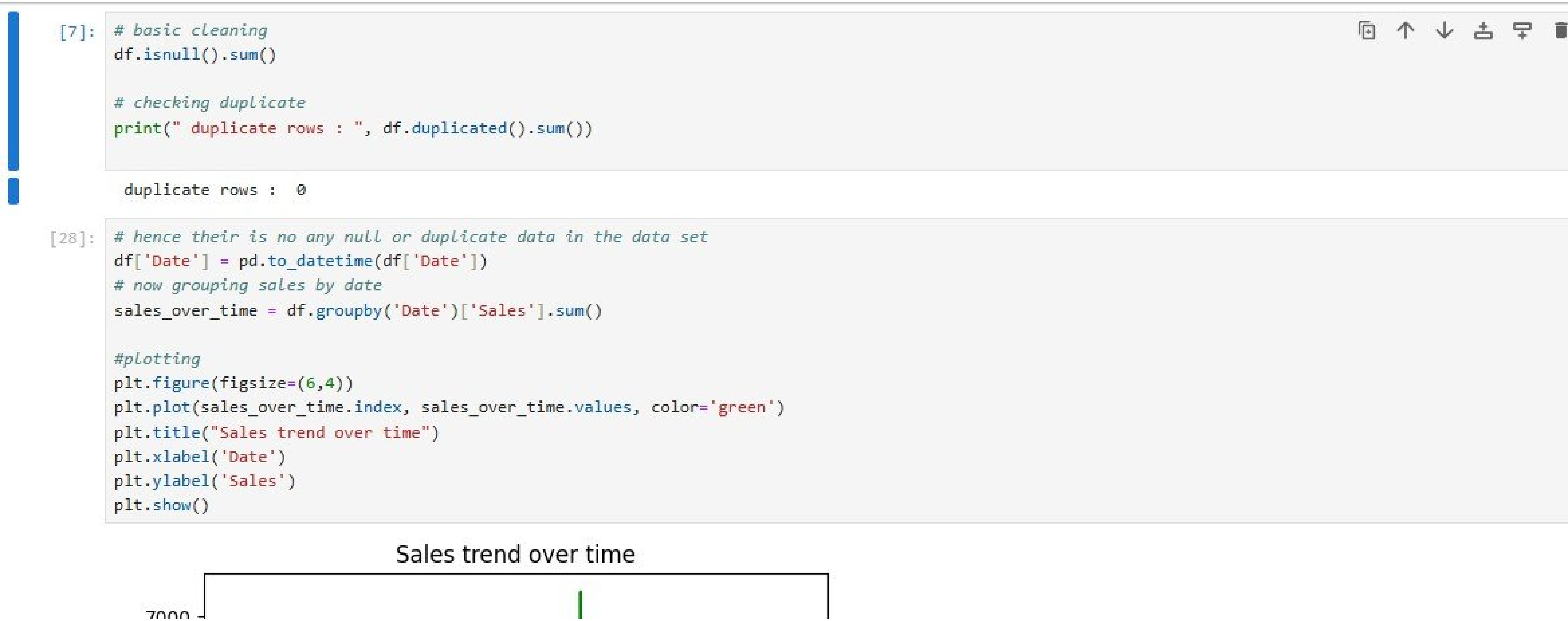


```
[5 rows x 24 columns]
```

```
[3]: # data cleaning start
```

```
[4]: # checking missing values
     df.isnull().sum()
```

```
[4]: Row ID              0
     Order ID            0
     Order Date          0
     Ship Date           0
     Ship Mode           0
     Customer ID         0
     Customer Name       0
     Segment             0
     City                0
     State               0
     Country             0
     Postal Code     41296
     Market              0
     Region              0
     Product ID          0
     Category            0
     Sub-Category        0
     Product Name        0
     Sales               0
     Quantity            0
     Discount            0
```



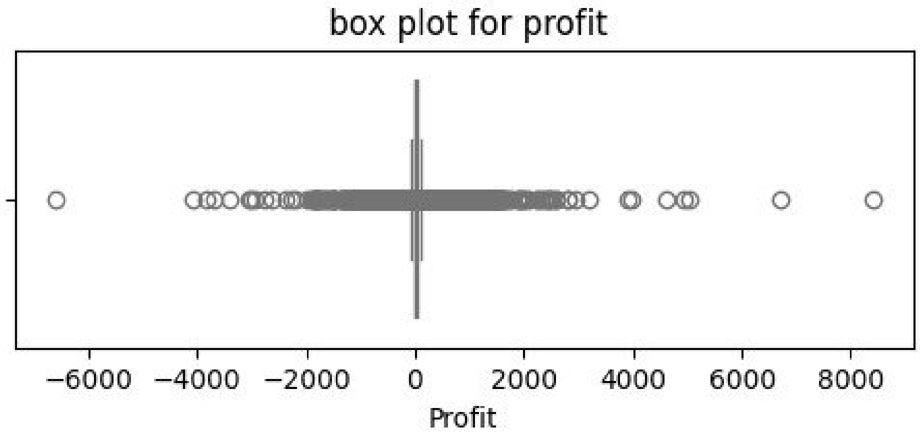
```
# basic cleaning
df.isnull().sum()

# checking duplicate
print(" duplicate rows : ", df.duplicated().sum())
```

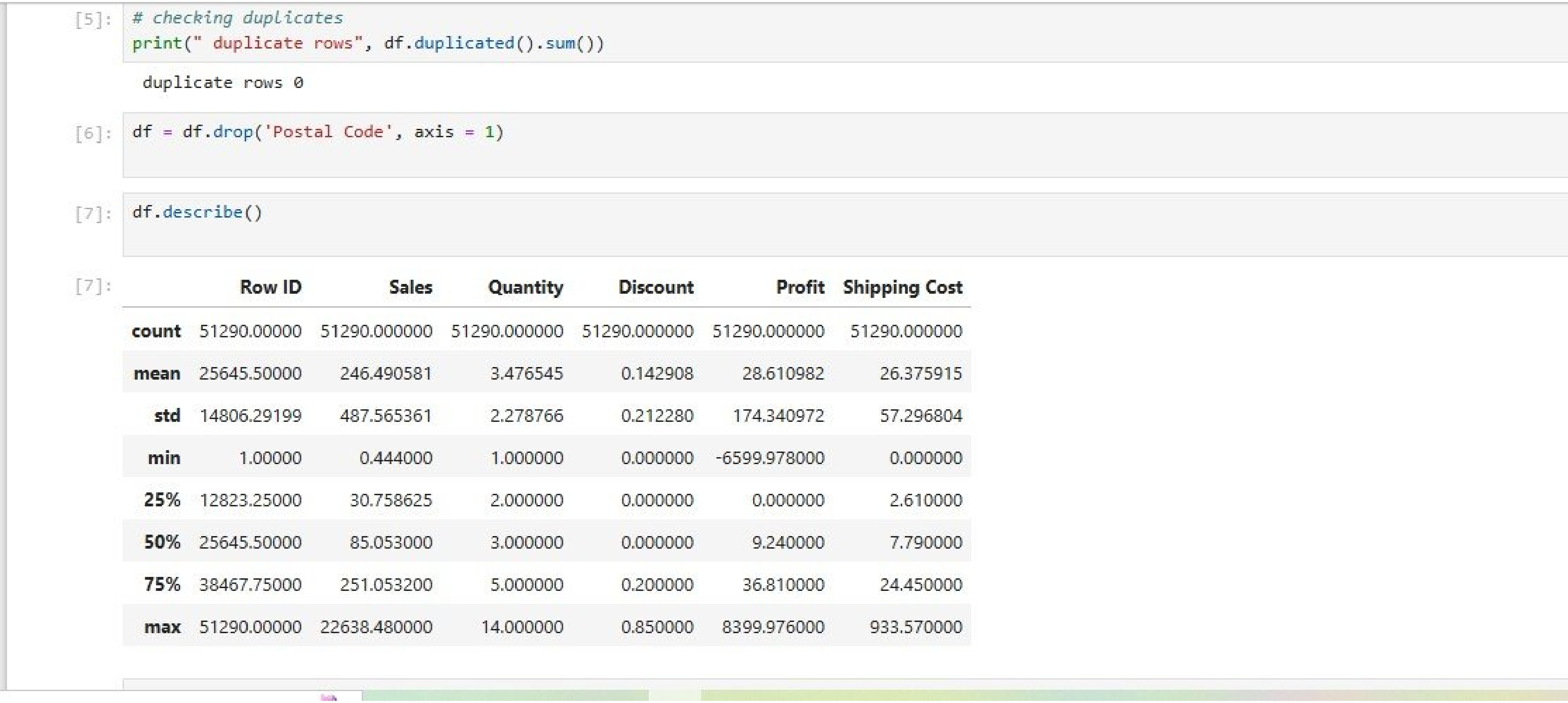
```
 duplicate rows :  0
```

```
# hence their is no any null or duplicate data in the data set
df['Date'] = pd.to_datetime(df['Date'])
# now grouping sales by date
sales_over_time = df.groupby('Date')['Sales'].sum()

#plotting
plt.figure(figsize=(6,4))
plt.plot(sales_over_time.index, sales_over_time.values, color='green')
plt.title("Sales trend over time")
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
```

```
[10]: # box plot  for Profit
      plt.figure(figsize=(6,2))
      sns.boxplot(x=df['Profit'], color='lightgreen')
      plt.title(" box plot for profit")
```

```
[10]: Text(0.5, 1.0, ' box plot for profit')
```





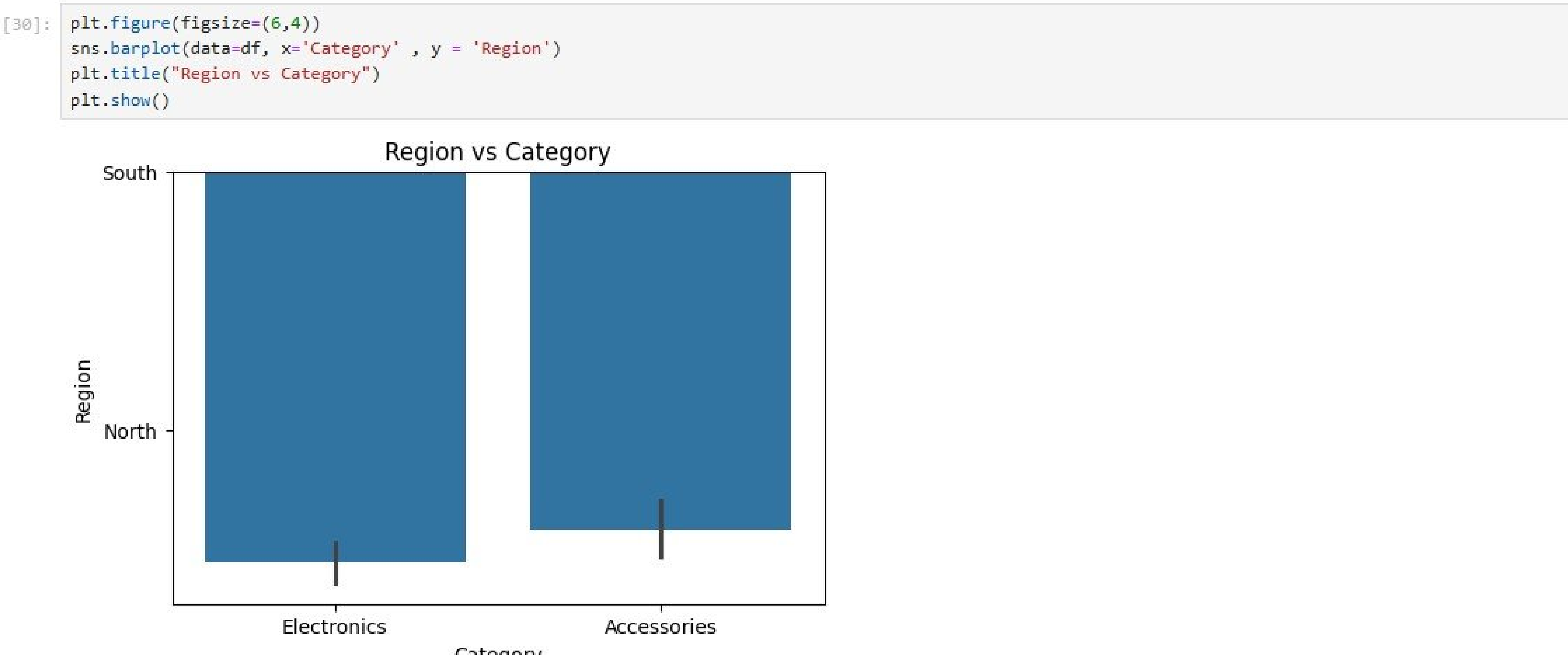
```
[5]: # checking duplicates
     print(" duplicate rows", df.duplicated().sum())

      duplicate rows 0
```

```
[6]: df = df.drop('Postal Code', axis = 1)
```

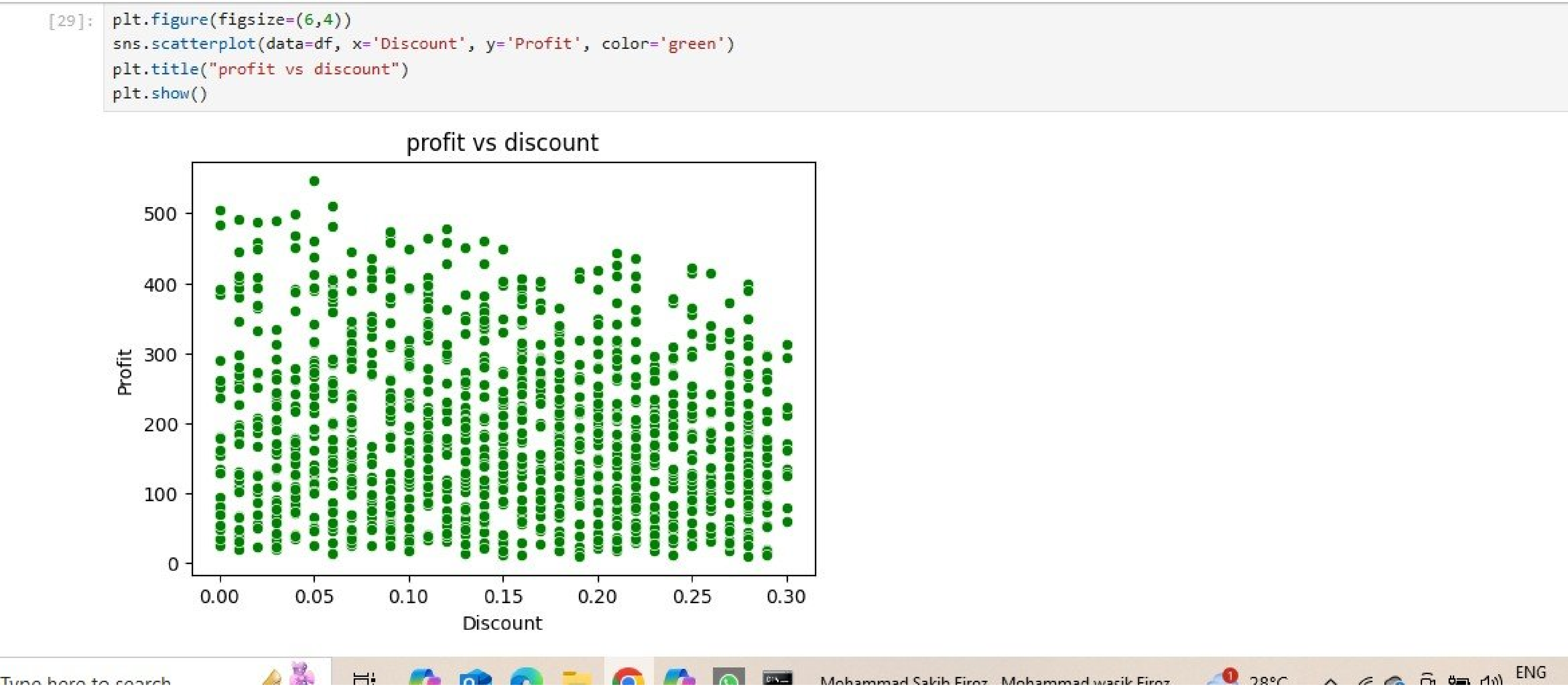
```
[7]: df.describe()
```

[7]:

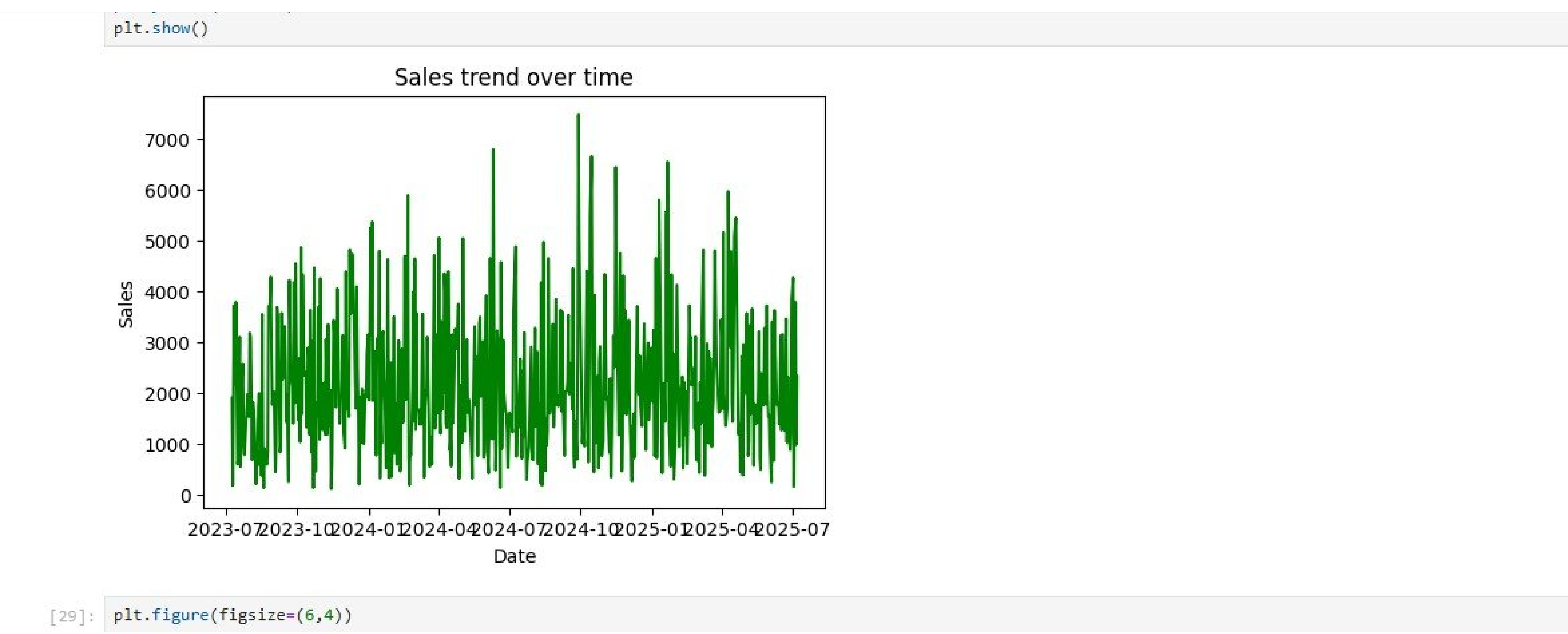
	Row ID	Sales	Quantity	Discount	Profit	Shipping Cost
count	51290.00000	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
mean	25645.50000	246.490581	3.476545	0.142908	28.610982	26.375915
std	14806.29199	487.565361	2.278766	0.212280	174.340972	57.296804
min	1.00000	0.444000	1.000000	0.000000	-6599.978000	0.000000
25%	12823.25000	30.758625	2.000000	0.000000	0.000000	2.610000
50%	25645.50000	85.053000	3.000000	0.000000	9.240000	7.790000
75%	38467.75000	251.053200	5.000000	0.200000	36.810000	24.450000
max	51290.00000	22638.480000	14.000000	0.850000	8399.976000	933.570000



```
[30]: plt.figure(figsize=(6,4))
      sns.barplot(data=df, x='Category' , y = 'Region')
      plt.title("Region vs Category")
      plt.show()
```

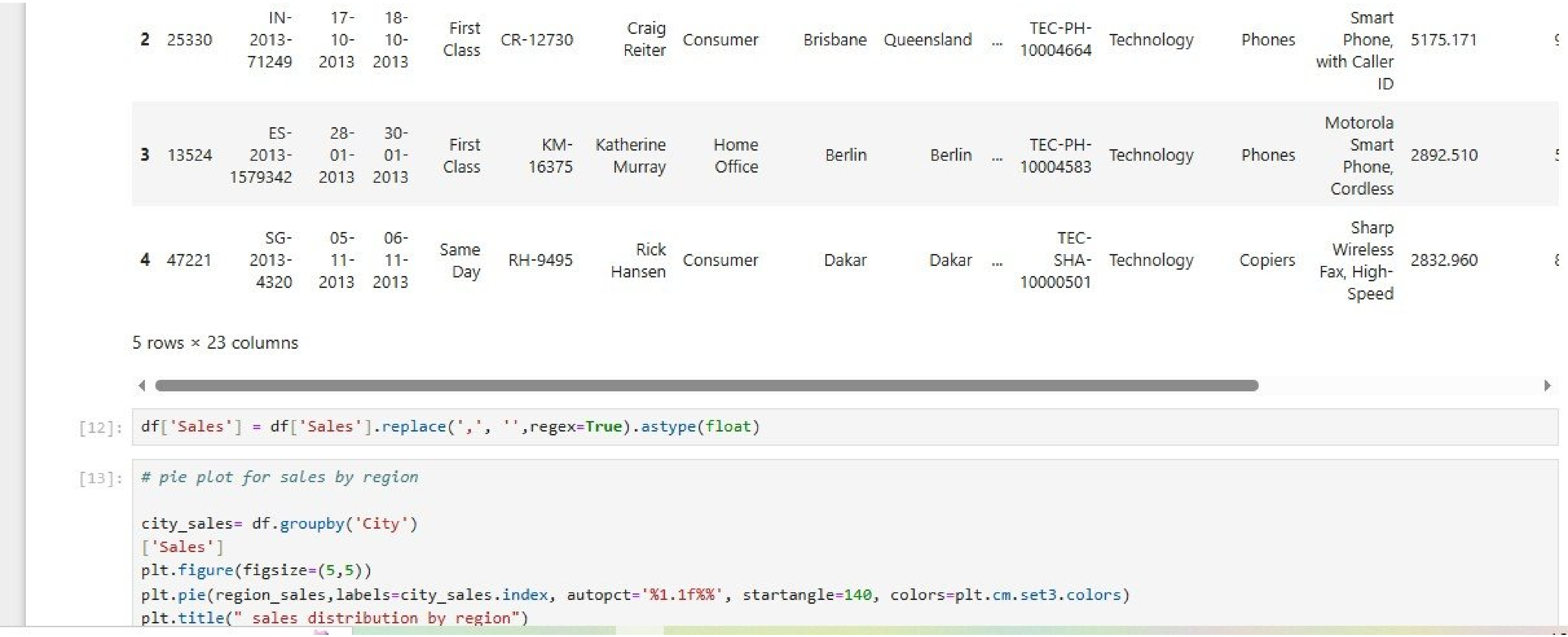


```
[29]: plt.figure(figsize=(6,4))
      sns.scatterplot(data=df, x='Discount', y='Profit', color='green')
      plt.title("profit vs discount")
      plt.show()
```



```
plt.show()
```

```
[29]: plt.figure(figsize=(6,4))
```

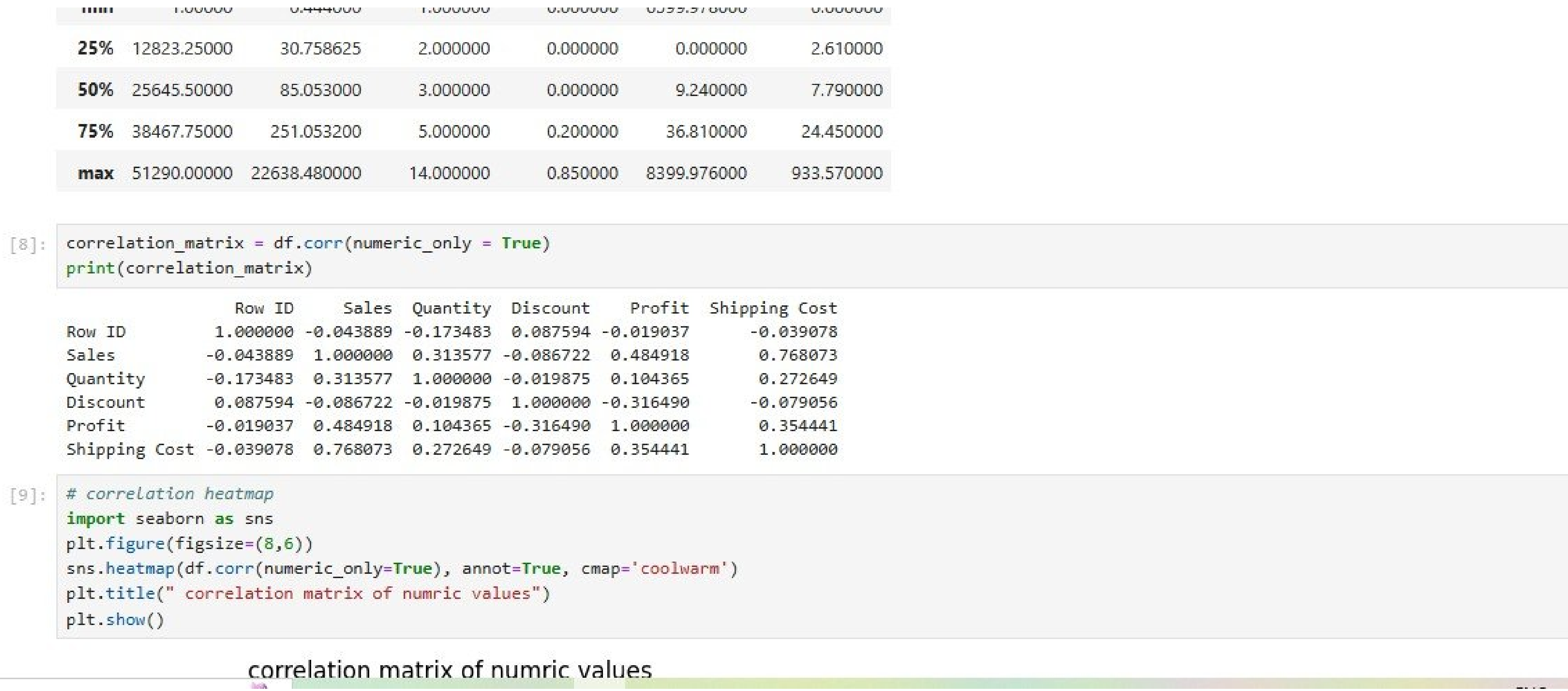


2	25330	IN-2013-71249	17-10-2013	18-10-2013	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	...	TEC-PH-10004664	Technology	Phones	Smart Phone, with Caller ID	5175.171
3	13524	ES-2013-1579342	28-01-2013	30-01-2013	First Class	KM-16375	Katherine Murray	Home Office	Berlin	Berlin	...	TEC-PH-10004583	Technology	Phones	Motorola Smart Phone, Cordless	2892.510
4	47221	SG-2013-4320	05-11-2013	06-11-2013	Same Day	RH-9495	Rick Hansen	Consumer	Dakar	Dakar	...	TEC-SHA-10000501	Technology	Copiers	Sharp Wireless Fax, High-Speed	2832.960

5 rows × 23 columns

[12]:
```
df['Sales'] = df['Sales'].replace(',', '',regex=True).astype(float)
```

[13]:
```
# pie plot for sales by region

city_sales= df.groupby('City')
['Sales']
plt.figure(figsize=(5,5))
plt.pie(region_sales,labels=city_sales.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.set3.colors)
plt.title(" sales distribution by region")
```



min	1.00000	0.444000	1.000000	0.000000	-6599.978000	0.000000
25%	12823.25000	30.758625	2.000000	0.000000	0.000000	2.610000
50%	25645.50000	85.053000	3.000000	0.000000	9.240000	7.790000
75%	38467.75000	251.053200	5.000000	0.200000	36.810000	24.450000
max	51290.00000	22638.480000	14.000000	0.850000	8399.976000	933.570000

[8]:

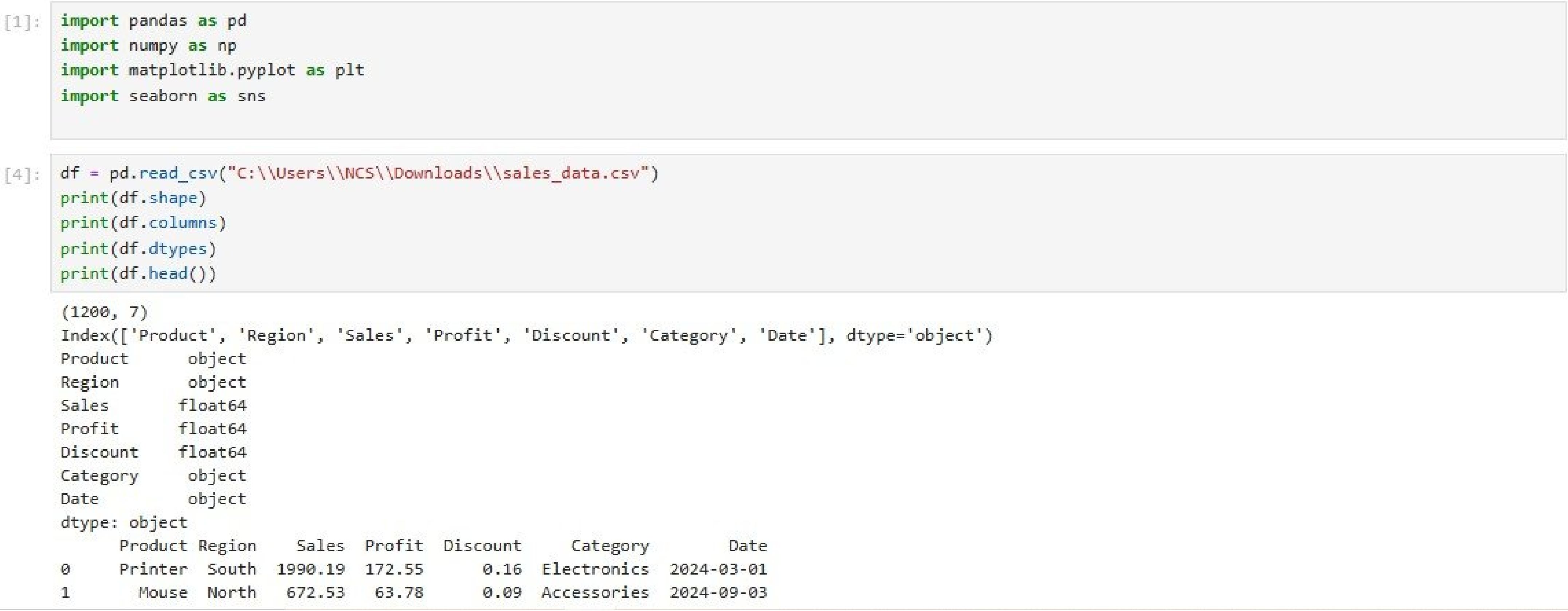
```
correlation_matrix = df.corr(numeric_only = True)
print(correlation_matrix)
```

```
                 Row ID     Sales  Quantity  Discount    Profit  Shipping Cost
Row ID         1.000000 -0.043889 -0.173483  0.087594 -0.019037      -0.039078
Sales         -0.043889  1.000000  0.313577 -0.086722  0.484918       0.768073
Quantity      -0.173483  0.313577  1.000000 -0.019875  0.104365       0.272649
Discount       0.087594 -0.086722 -0.019875  1.000000 -0.316490      -0.079056
Profit        -0.019037  0.484918  0.104365 -0.316490  1.000000       0.354441
Shipping Cost -0.039078  0.768073  0.272649 -0.079056  0.354441       1.000000
```

[9]:

```
# correlation heatmap
import seaborn as sns
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title(" correlation matrix of numric values")
plt.show()
```

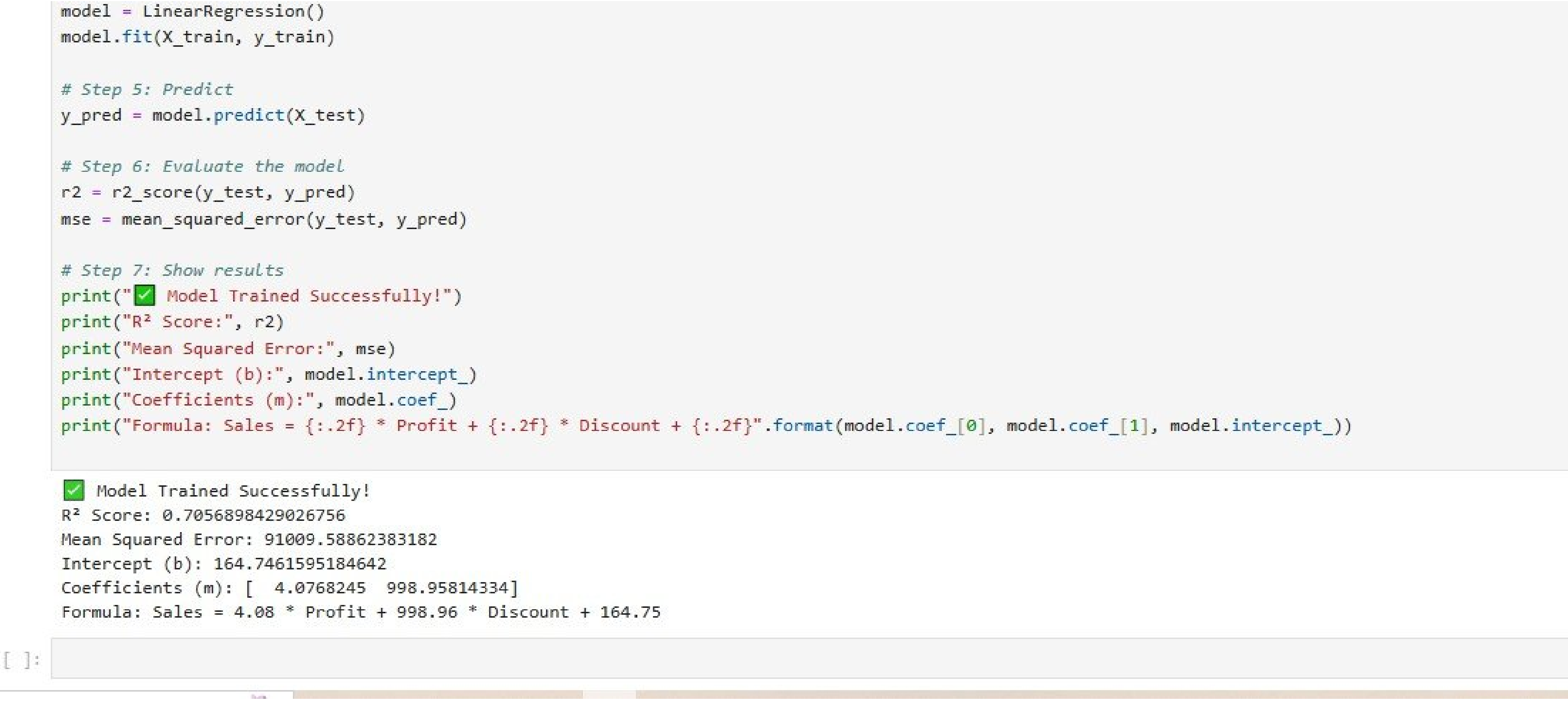
correlation matrix of numric values



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
df = pd.read_csv("C:\\Users\\NCS\\Downloads\\sales_data.csv")
print(df.shape)
print(df.columns)
print(df.dtypes)
print(df.head())
```

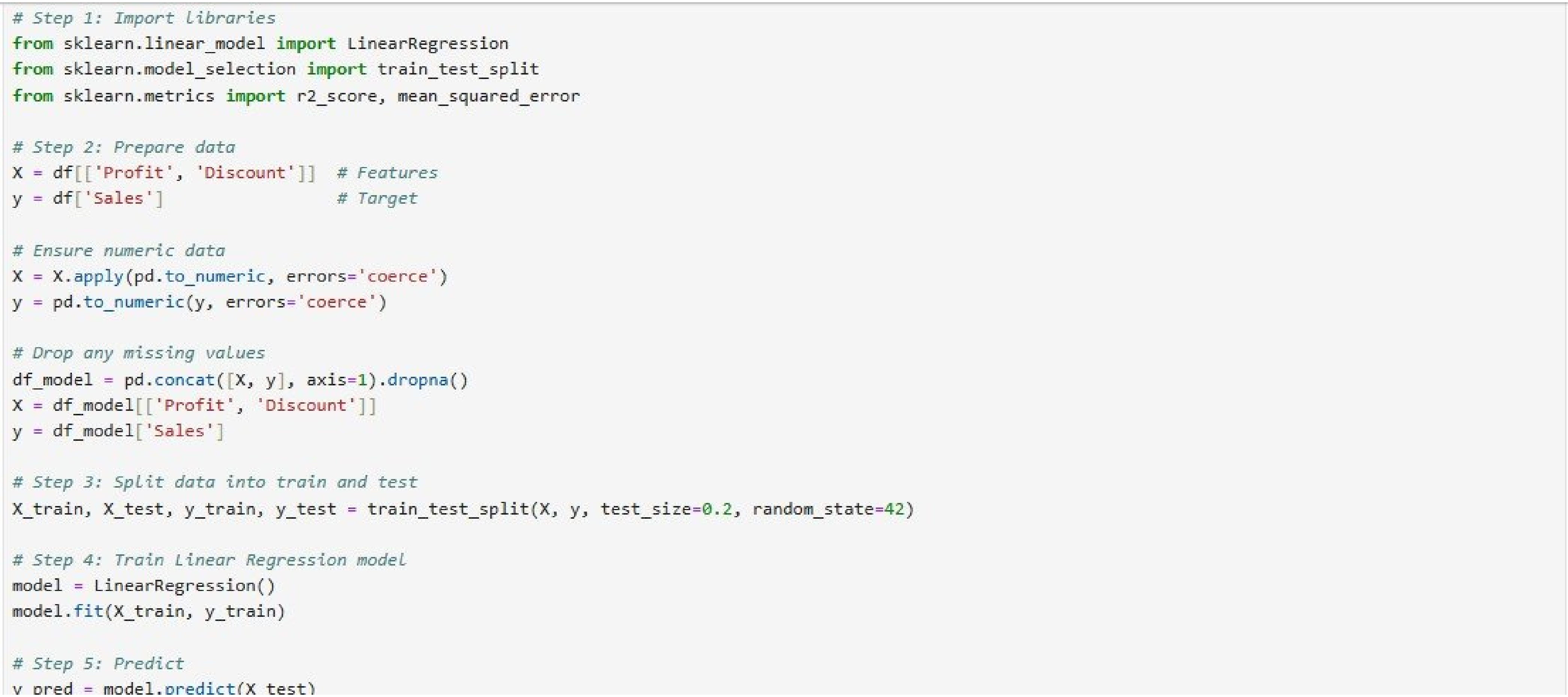
```
(1200, 7)
Index(['Product', 'Region', 'Sales', 'Profit', 'Discount', 'Category', 'Date'], dtype='object')
Product      object
Region       object
Sales       float64
Profit      float64
Discount    float64
Category     object
Date         object
dtype: object
      Product Region    Sales  Profit  Discount     Category        Date
0     Printer  South  1990.19  172.55      0.16  Electronics  2024-03-01
1       Mouse  North   672.53   63.78      0.09  Accessories  2024-09-03
```



```
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Predict
y_pred = model.predict(X_test)

# Step 6: Evaluate the model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

# Step 7: Show results
print("✅ Model Trained Successfully!")
print("R² Score:", r2)
print("Mean Squared Error:", mse)
print("Intercept (b):", model.intercept_)
print("Coefficients (m):", model.coef_)
print("Formula: Sales = {:.2f} * Profit + {:.2f} * Discount + {:.2f}".format(model.coef_[0], model.coef_[1], model.intercept_))
```

```
✅ Model Trained Successfully!
R² Score: 0.7056898429026756
Mean Squared Error: 91009.58862383182
Intercept (b): 164.7461595184642
Coefficients (m): [  4.0768245  998.95814334]
Formula: Sales = 4.08 * Profit + 998.96 * Discount + 164.75
```

```
[ ]:
```



```
# Step 1: Import libraries
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error

# Step 2: Prepare data
X = df[['Profit', 'Discount']]  # Features
y = df['Sales']                 # Target

# Ensure numeric data
X = X.apply(pd.to_numeric, errors='coerce')
y = pd.to_numeric(y, errors='coerce')

# Drop any missing values
df_model = pd.concat([X, y], axis=1).dropna()
X = df_model[['Profit', 'Discount']]
y = df_model['Sales']

# Step 3: Split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Predict
y_pred = model.predict(X_test)
```