In [1]:

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

df=pd.read_csv("Supermart Grocery Sales.csv")
df.head(5)
```

Out[1]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	0.12
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	0.18
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	0.21
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	0.25
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	0.26

In [2]:

```python
df.tail(5)
```

Out[2]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Disco
9989	OD9990	Sudeep	Eggs, Meat & Fish	Eggs	Madurai	12/24/2015	West	945	0
9990	OD9991	Alan	Bakery	Biscuits	Kanyakumari	07-12-2015	West	1195	0
9991	OD9992	Ravi	Food Grains	Rice	Bodi	06-06-2017	West	1567	0
9992	OD9993	Peer	Oil & Masala	Spices	Pudukottai	10/16/2018	West	1659	0
9993	OD9994	Ganesh	Food Grains	Atta & Flour	Tirunelveli	4/17/2018	West	1034	0

In [3]:

```python
df.shape
```

Out[3]:

```
(9994, 11)
```

In [4]:

```python
print("no of rows",df.shape[0])
```

```
no of rows 9994
```

In [5]:

```python
print("no of columns",df.shape[1])
```

```
no of columns 11
```

In [6]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Order ID       9994 non-null   object 
 1   Customer Name  9994 non-null   object 
 2   Category       9994 non-null   object 
 3   Sub Category   9994 non-null   object 
 4   City           9994 non-null   object 
 5   Order Date     9994 non-null   object 
 6   Region         9994 non-null   object 
 7   Sales          9994 non-null   int64  
 8   Discount       9994 non-null   float64
 9   Profit         9994 non-null   float64
 10  State          9994 non-null   object 
dtypes: float64(2), int64(1), object(8)
memory usage: 859.0+ KB
```

In [7]:

```
df.describe()
```

Out[7]:

	Sales	Discount	Profit
count	9994.000000	9994.000000	9994.000000
mean	1496.596158	0.226817	374.937082
std	577.559036	0.074636	239.932881
min	500.000000	0.100000	25.250000
25%	1000.000000	0.160000	180.022500
50%	1498.000000	0.230000	320.780000
75%	1994.750000	0.290000	525.627500
max	2500.000000	0.350000	1120.950000

In [8]:

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

Out[8]:

```
Order ID         0
Customer Name    0
Category         0
Sub Category     0
City             0
Order Date       0
Region           0
Sales            0
Discount         0
Profit           0
State            0
dtype: int64
```

In [10]:

```
# Check for duplicates
df.drop_duplicates(inplace=True)
```

2. Convert Date Columns to DateTime Format

In [11]:

```
df.columns
```

Out[11]:

```
Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
       'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State'],
      dtype='object')
```

In [40]:

```
print("original type of date is=",df['Order Date'].dtype)
```

```
original type of date is= datetime64[ns]
```

In [41]:

```
print("after convert type of date is=",df['new_Date'].dtype)
```

```
after convert type of date is= object
```

In [28]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Order ID       9994 non-null   object
 1   Customer Name  9994 non-null   object
 2   Category       9994 non-null   object
 3   Sub Category   9994 non-null   object
 4   City           9994 non-null   object
 5   Order Date     9994 non-null   object
 6   Region         9994 non-null   object
 7   Sales          9994 non-null   int64 
 8   Discount       9994 non-null   float64
 9   Profit         9994 non-null   float64
 10  State          9994 non-null   object
 11  new_Date       9994 non-null   object
dtypes: float64(2), int64(1), object(9)
memory usage: 937.1+ KB
```

In [37]:

```
 df['Order Date'] = pd.to_datetime(df['Order Date'], format='%Y-%m-%d',errors='c
```

In [38]:

```
print(df['Order Date'].dtype)
```

```
datetime64[ns]
```

In [39]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Order ID       9994 non-null   object        
 1   Customer Name  9994 non-null   object        
 2   Category       9994 non-null   object        
 3   Sub Category   9994 non-null   object        
 4   City           9994 non-null   object        
 5   Order Date     0 non-null      datetime64[ns]
 6   Region         9994 non-null   object        
 7   Sales          9994 non-null   int64         
 8   Discount       9994 non-null   float64       
 9   Profit         9994 non-null   float64       
 10  State          9994 non-null   object        
 11  new_Date       9994 non-null   object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(8)
memory usage: 937.1+ KB
```

In [42]:

```
# find the total sale by category
df.columns
```

Out[42]:

```
Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
       'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State',
       'new_Date'],
      dtype='object')
```

In [43]:

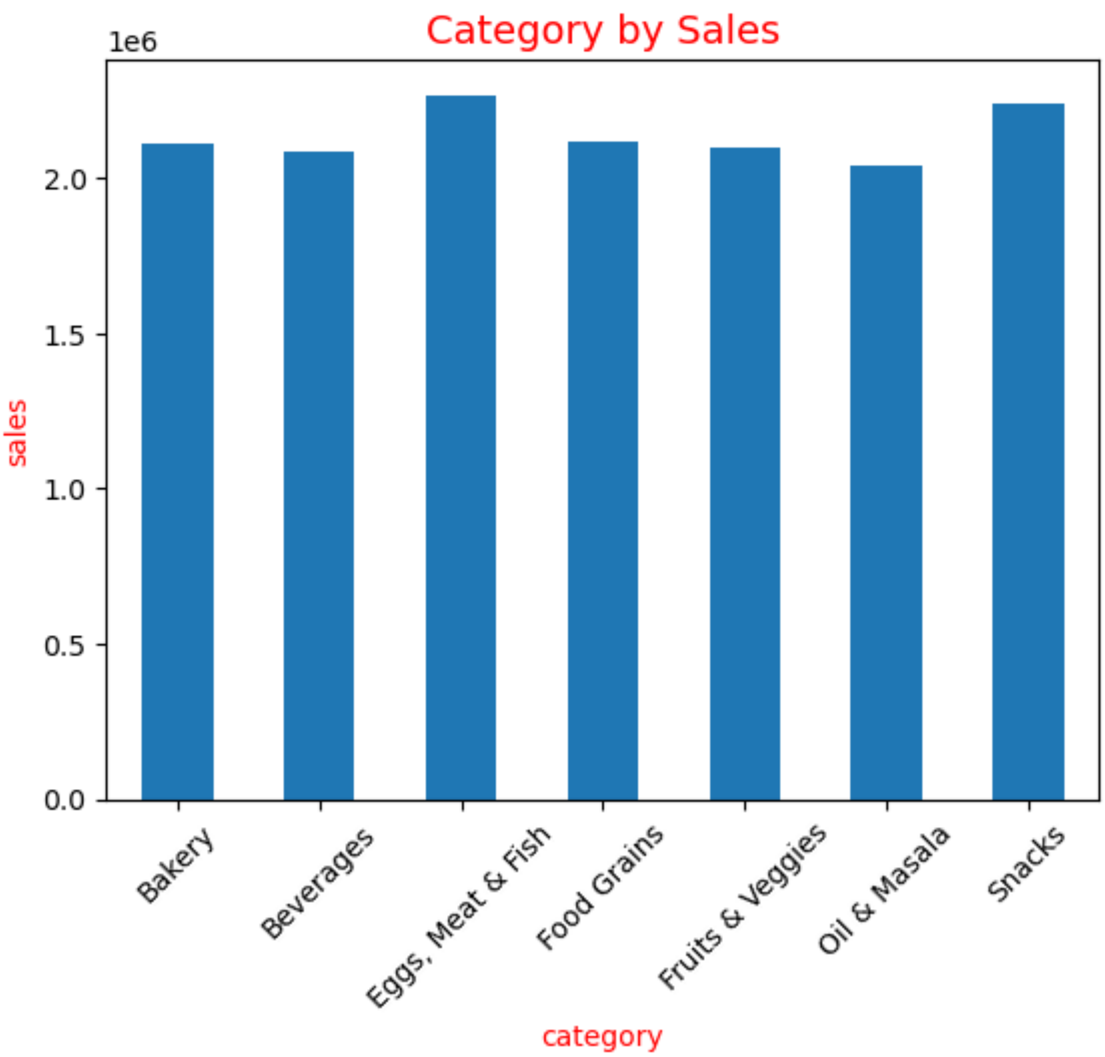
```
sales_category=df.groupby('Category')["Sales"].sum()
sales_category
```

Out[43]:

```
Category
Bakery               2112281
Beverages            2085313
Eggs, Meat & Fish    2267401
Food Grains          2115272
Fruits & Veggies     2100727
Oil & Masala         2038442
Snacks               2237546
Name: Sales, dtype: int64
```

In [46]:

```
# create a bar plot using sales_category
sales_category.plot(kind='bar')
plt.title("Category by Sales", fontsize = 14,color="red")
plt.xlabel("category",fontsize=10,color="red")
plt.ylabel("sales",fontsize=10,color="red")
plt.xticks(rotation=45)
plt.show()
```



```
In [ ]: # from above bar graph we can conclude that eggs,Meat & Fish have the higest sal
```

```
In [50]: # Extract day,month,year  from the order date
         df['Order Day'] = df['Order Date'].dt.day
         df['Order Month'] = df['Order Date'].dt.month
         df['Order Year'] = df['Order Date'].dt.year
```

```
In [51]: # sum of sales by year
         df.columns
```

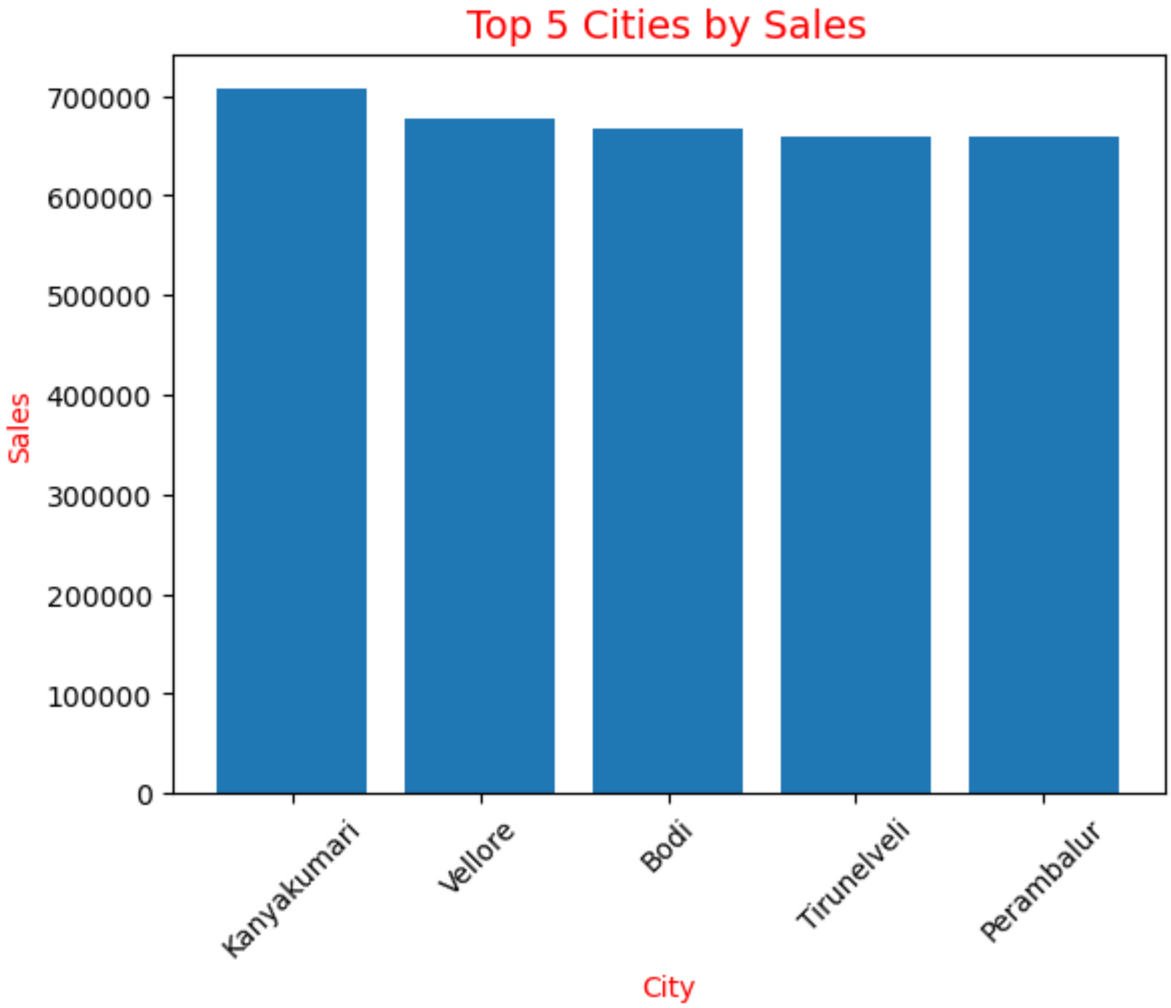
```
Out[51]: Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
                'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State',
                'new_Date', 'month_no', 'Order Day', 'Order Month', 'Order Year'],
               dtype='object')
```

```
In [3]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns

        df=pd.read_csv("Supermart Grocery Sales.csv")

        # step 1extract the relevent columns
        city_sales=df[['City','Sales']]
        # Step 2: Calculate total sales per city
        total_sales=city_sales.groupby('City').sum()
```

```
# Step 3: Sort the cities by sales
sorted_cities=total_sales.sort_values(by="Sales",ascending=False)
# Step 4: Select the top 5 cities
top_5=sorted_cities.head(5)
# Step 5: Plot the bar chart
plt.bar(top_5.index, top_5['Sales'])
plt.xlabel('City',fontsize=10,color="r")
plt.ylabel('Sales',fontsize=10,color="r")
plt.title('Top 5 Cities by Sales' ,fontsize=14,color="r")
plt.xticks(rotation=45)
plt.show()
```



3. Label Encoding for Categorical Variables

In [4]:

```
#Convert categorical   such as Category, Sub Category, City, Region,
#State, and Month into numerical values.
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score

# encode categorical variables
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
df=pd.read_csv("Supermart Grocery Sales.csv")
df.columns
```

Out[4]:

```
Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
       'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State'],
      dtype='object')
```

In [6]:

```
# Initialize the label encoder
le = LabelEncoder()


df['Category']=le.fit_transform(df['Category'])
df['Sub Category']=le.fit_transform(df['Sub Category'])
df['City']=le.fit_transform(df['City'])
df['Region']=le.fit_transform(df['Region'])
df['State']=le.fit_transform(df['State'])
```

In [7]:

```
# after encoding showing the first 5 rows of dataframe
df.head(5)
```

Out[7]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profi
0	OD1	Harish	5	14	21	11-08-2017	2	1254	0.12	401.2
1	OD2	Sudha	1	13	8	11-08-2017	3	749	0.18	149.8
2	OD3	Hussain	3	0	13	06-12-2017	4	2360	0.21	165.2
3	OD4	Jackson	4	12	4	10-11-2016	3	896	0.25	89.6
4	OD5	Ridhesh	3	18	12	10-11-2016	3	2355	0.26	918.4

Step 4: Exploratory Data Analysis (EDA)

In [8]:

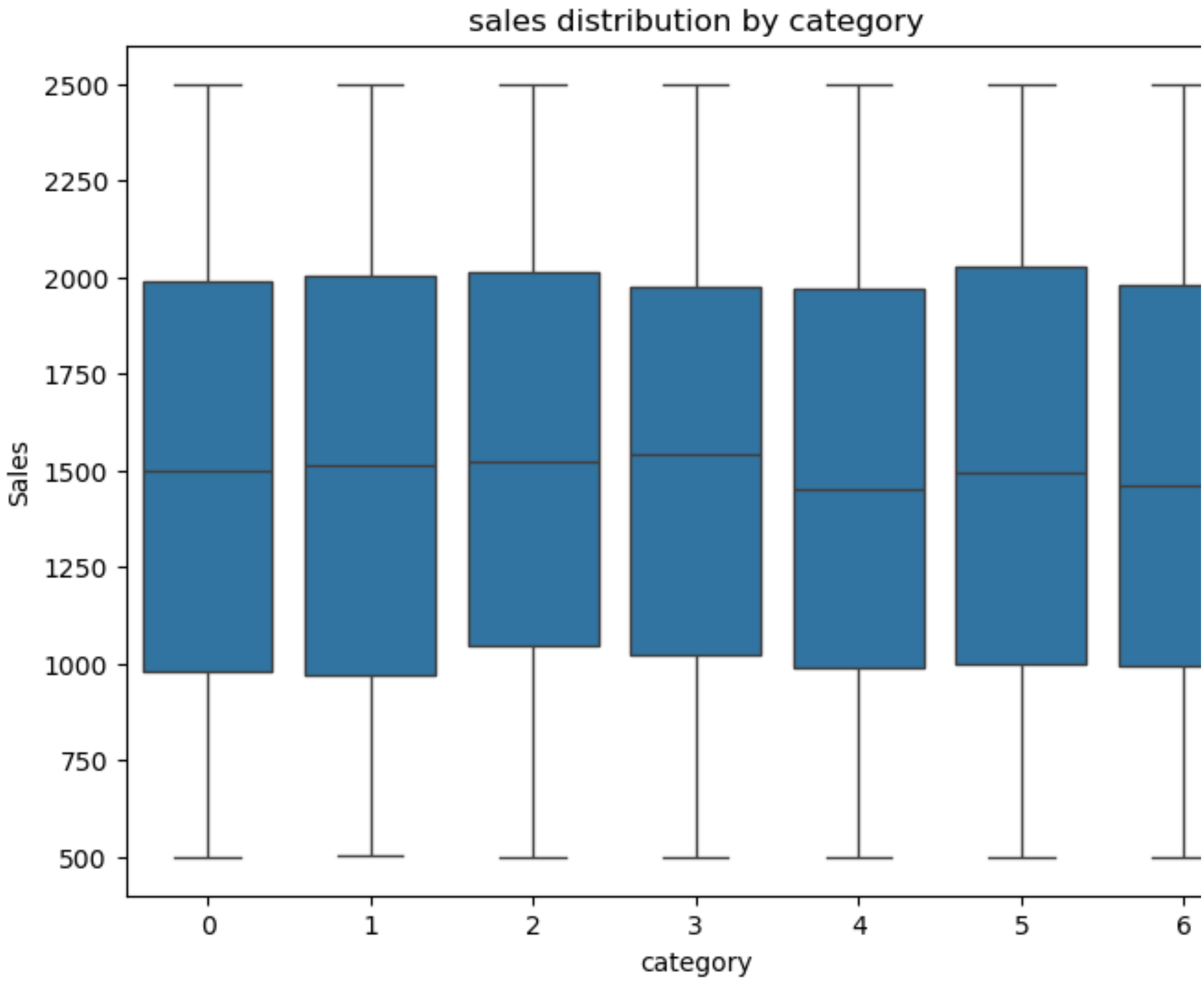
```
# create a box plot using sales_category
#  Distribution of Sales by Category
df.columns
```

Out[8]:

```
Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
       'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State'],
      dtype='object')
```

In [24]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8,6))
sns.boxplot(x='Category',y='Sales',data=df)
plt.title("sales distribution by category")
plt.xlabel("category")
plt.ylabel("Sales")
plt.show()
```



2. Sales Trends Over Time

In [11]:

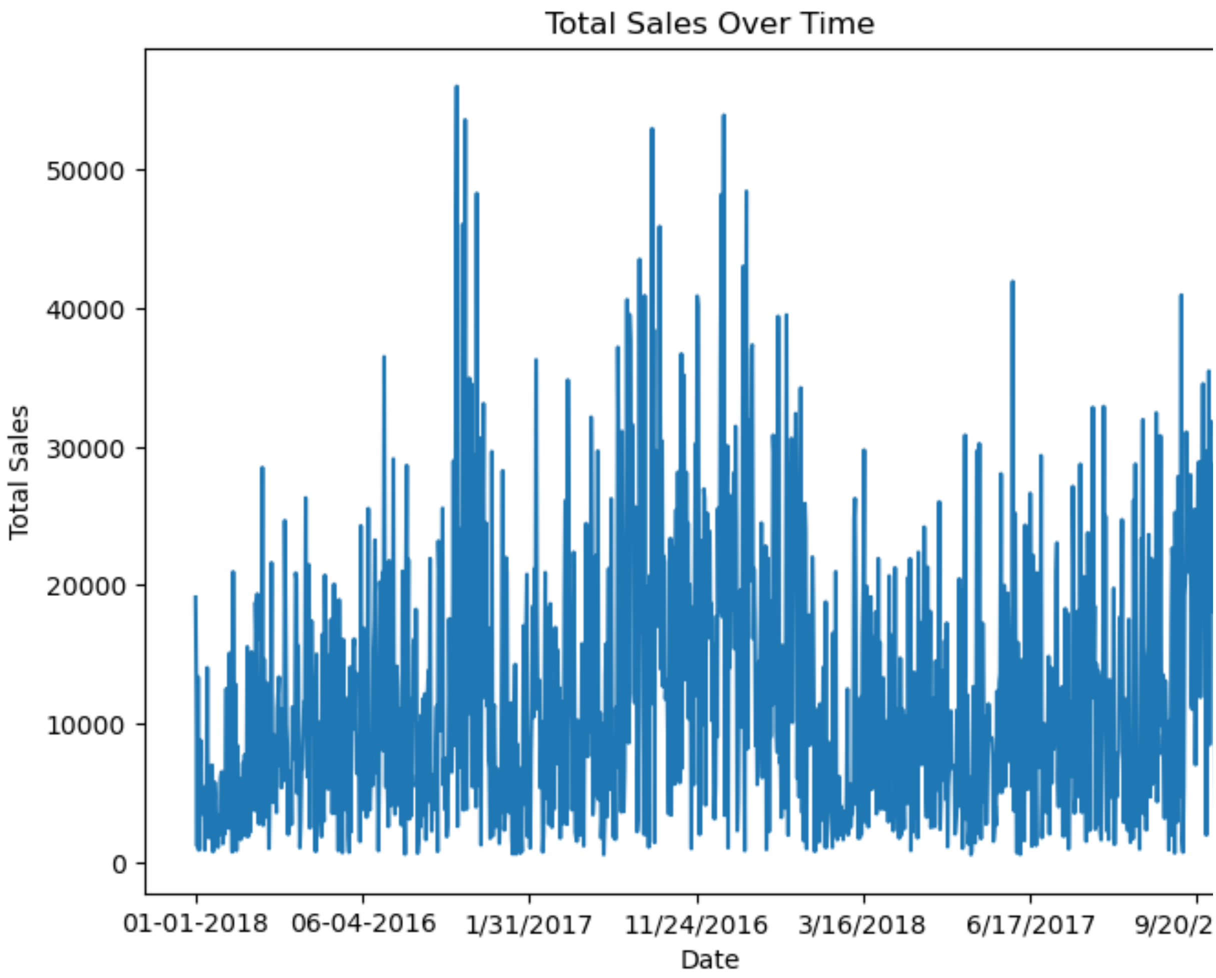
```
df.columns
```

Out[11]:

```
Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
       'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State'],
      dtype='object')
```

In [13]:

```
plt.figure(figsize=(8,6))
df.groupby('Order Date')['Sales'].sum().plot()
plt.title('Total Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()
```



Step 5: Feature Selection and Model Building

```
In [15]: # select features and target columns
         df.columns
```

```
Out[15]: Index(['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City',
                'Order Date', 'Region', 'Sales', 'Discount', 'Profit', 'State'],
               dtype='object')
```

```
In [18]: # in the above columns we use feature matrix variable'Category', 'Sub Category',
         # and reaming columns we will drop
         feature_data=df.drop(columns=['Order ID','Customer Name','Order Date','Sales'])
```

```
In [19]: target_data=df['Sales']
```

```
In [ ]:
```

```
In [22]: # now split the data in train and test
         x_train,x_test,y_train,y_test=train_test_split(feature_data,target_data,test_siz
         # feautre_sacling
         scaler = StandardScaler()
         x_train=scaler.fit_transform(x_train)
         x_test=scaler.transform(x_test)
```

Step 6: Train a Linear Regression Model

```
In [23]: # Initialize the model
         model=LinearRegression()
```

```
# train the model
model.fit(x_train,y_train)
# make prediction
pred=model.predict(x_test)
```

Step 7: Evaluate the Model

In [26]:

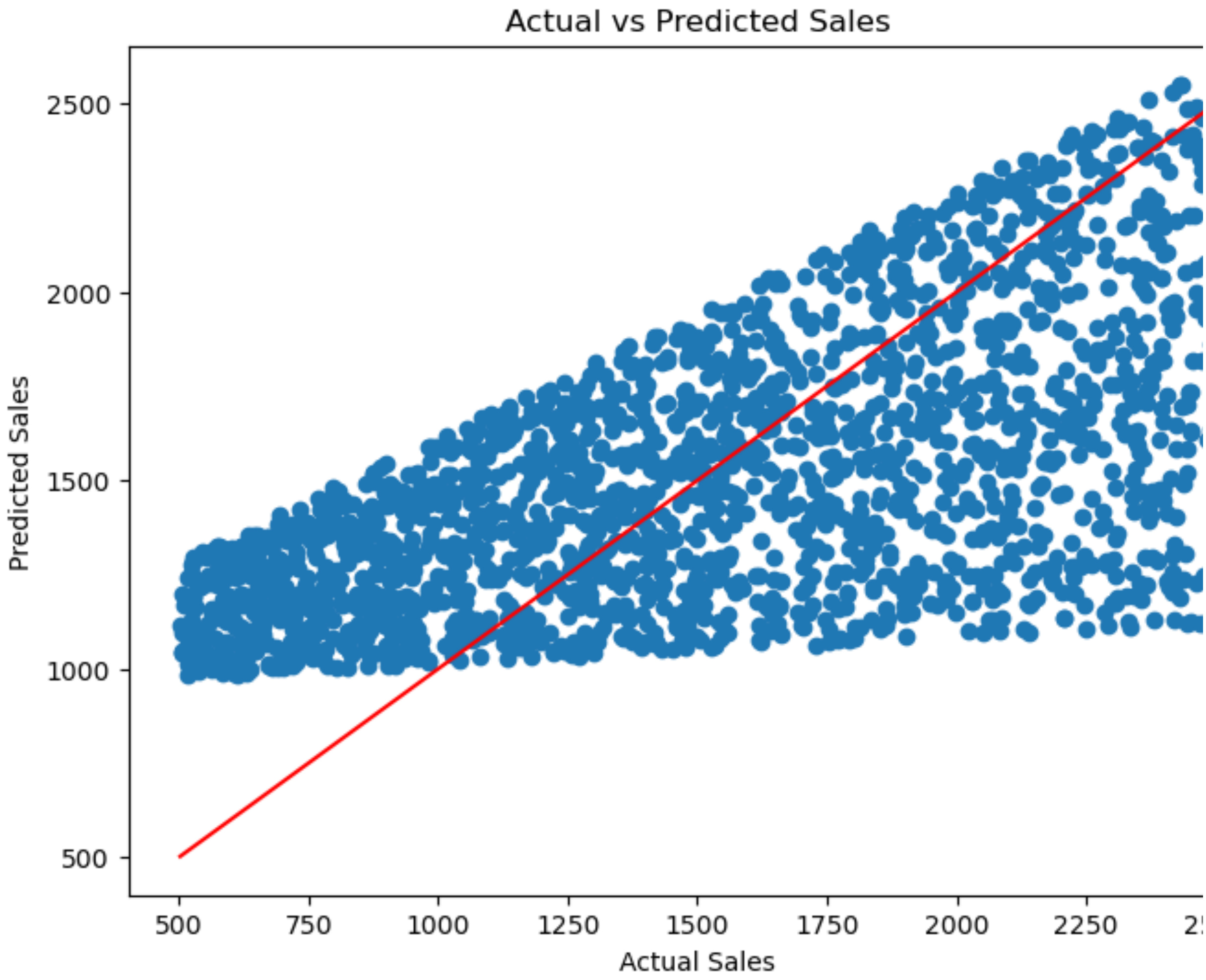
```
# Evaluate the model performance using Mean Squared Error (MSE) and R-squared
# calculate mean square error and r-squared
mse = mean_squared_error(y_test, pred)
r2 = r2_score(y_test, pred)
print(f"mean_squared_error:{mse}")
print(f"r_squared:{r2}")
```

```
mean_squared_error:209945.1618047447
r_squared:0.3661838539425728
```

Step 8: Visualize the Results

In [27]:

```
# Actual vs Predicted Sales
plt.figure(figsize=(8, 6))
plt.scatter(y_test, pred)
plt.plot([min(y_test), max(y_test)], [min(y_test),max(y_test)], color='red')
plt.title('Actual vs Predicted Sales')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.show()
```



In [ ]: