In [3]:
```python
import pandas as pd
```

In [5]:
```python
Amason_sales_data = pd.read_csv(r"C:\Users\steph\OneDrive\Documents\Desktop\Amazon sales data\amazon_sales_data 2025.csv")
```

In [7]:
```python
Amason_sales_data.head()
```

Out[7]:

	Order ID	Date	Product	Category	Price	Quantity	Total Sales	Customer Name	Customer Location	Payment Method	Status
0	ORD0001	14-03-2025	Running Shoes	Footwear	60	3	180	Emma Clark	New York	Debit Card	Cancelled
1	ORD0002	20-03-2025	Headphones	Electronics	100	4	400	Emily Johnson	San Francisco	Debit Card	Pending
2	ORD0003	15-02-2025	Running Shoes	Footwear	60	2	120	John Doe	Denver	Amazon Pay	Cancelled
3	ORD0004	19-02-2025	Running Shoes	Footwear	60	3	180	Olivia Wilson	Dallas	Credit Card	Pending
4	ORD0005	10-03-2025	Smartwatch	Electronics	150	3	450	Emma Clark	New York	Debit Card	Pending

In [9]:
```python
Amason_sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Order ID           250 non-null    object
 1   Date               250 non-null    object
 2   Product            250 non-null    object
 3   Category           250 non-null    object
 4   Price              250 non-null    int64
 5   Quantity           250 non-null    int64
 6   Total Sales        250 non-null    int64
 7   Customer Name      250 non-null    object
 8   Customer Location  250 non-null    object
 9   Payment Method     250 non-null    object
 10  Status             250 non-null    object
dtypes: int64(3), object(8)
memory usage: 21.6+ KB
```

In [11]:
```python
## Checking the correct Date format
```

In [13]:
```python
Amason_sales_data["Date"] = pd.to_datetime(Amason_sales_data["Date"],format="%d-%m-%Y")
```

In [15]:
```python
## Changing the column name
```

In [17]:
```python
Amason_sales_data.rename(columns={Amason_sales_data.columns[1]:"Order_Date"},inplace=True)
```

In [19]:
```python
Amason_sales_data["Order_Date"] = pd.to_datetime(Amason_sales_data["Order_Date"],errors = "coerce")
```

In [21]:
```python
Amason_sales_data["Order_Date"].head()
```

Out[21]:
```
0   2025-03-14
1   2025-03-20
2   2025-02-15
3   2025-02-19
4   2025-03-10
Name: Order_Date, dtype: datetime64[ns]
```

for checking that product name and all in correct space or extra space

if its had extra space we need to remove them

In [27]:
```python
Amason_sales_data.apply(lambda x:x.str.strip() if x.dtype== "object" else x)
```

Out[27]:

	Order ID	Order_Date	Product	Category	Price	Quantity	Total Sales	Customer Name	Customer Location	Payment Method	Status
0	ORD0001	2025-03-14	Running Shoes	Footwear	60	3	180	Emma Clark	New York	Debit Card	Cancelled
1	ORD0002	2025-03-20	Headphones	Electronics	100	4	400	Emily Johnson	San Francisco	Debit Card	Pending
2	ORD0003	2025-02-15	Running Shoes	Footwear	60	2	120	John Doe	Denver	Amazon Pay	Cancelled
3	ORD0004	2025-02-19	Running Shoes	Footwear	60	3	180	Olivia Wilson	Dallas	Credit Card	Pending
4	ORD0005	2025-03-10	Smartwatch	Electronics	150	3	450	Emma Clark	New York	Debit Card	Pending
...	...	...	...	...	...	...	...	...	...	...	...
245	ORD0246	2025-03-17	T-Shirt	Clothing	20	2	40	Daniel Harris	Miami	Debit Card	Cancelled
246	ORD0247	2025-03-30	Jeans	Clothing	40	1	40	Sophia Miller	Dallas	Debit Card	Cancelled
247	ORD0248	2025-03-05	T-Shirt	Clothing	20	2	40	Chris White	Denver	Debit Card	Cancelled
248	ORD0249	2025-03-08	Smartwatch	Electronics	150	3	450	Emily Johnson	New York	Debit Card	Cancelled
249	ORD0250	2025-02-19	Smartphone	Electronics	500	4	2000	Emily Johnson	Seattle	Amazon Pay	Completed

250 rows × 11 columns

In [29]:
```python
Amason_sales_data.columns = Amason_sales_data.columns.str.strip().str.replace(" ","_") # for column name
```

In [31]:
```python
Amason_sales_data.columns
```

Out[31]:
```
Index(['Order_ID', 'Order_Date', 'Product', 'Category', 'Price', 'Quantity',
       'Total_Sales', 'Customer_Name', 'Customer_Location', 'Payment_Method',
       'Status'],
      dtype='object')
```

In [33]:
```python
Amason_sales_data.dropna(subset=['Order_Date', 'Price', 'Quantity', 'Total_Sales'], inplace=True)
```

In [35]:
```python
Amason_sales_data.to_csv("clean Data_csv",index=False) # this data  for PowerBI Analysis
```

In [37]:
```python
## Exploratory Data Analysis (EDA)
```

In [39]:
```python
import pandas as pd
```

In [41]:
```python
import matplotlib.pyplot as plt
```

In [42]:
```python
import seaborn as sns
```

In [43]:
```python
## best performing product categories
```

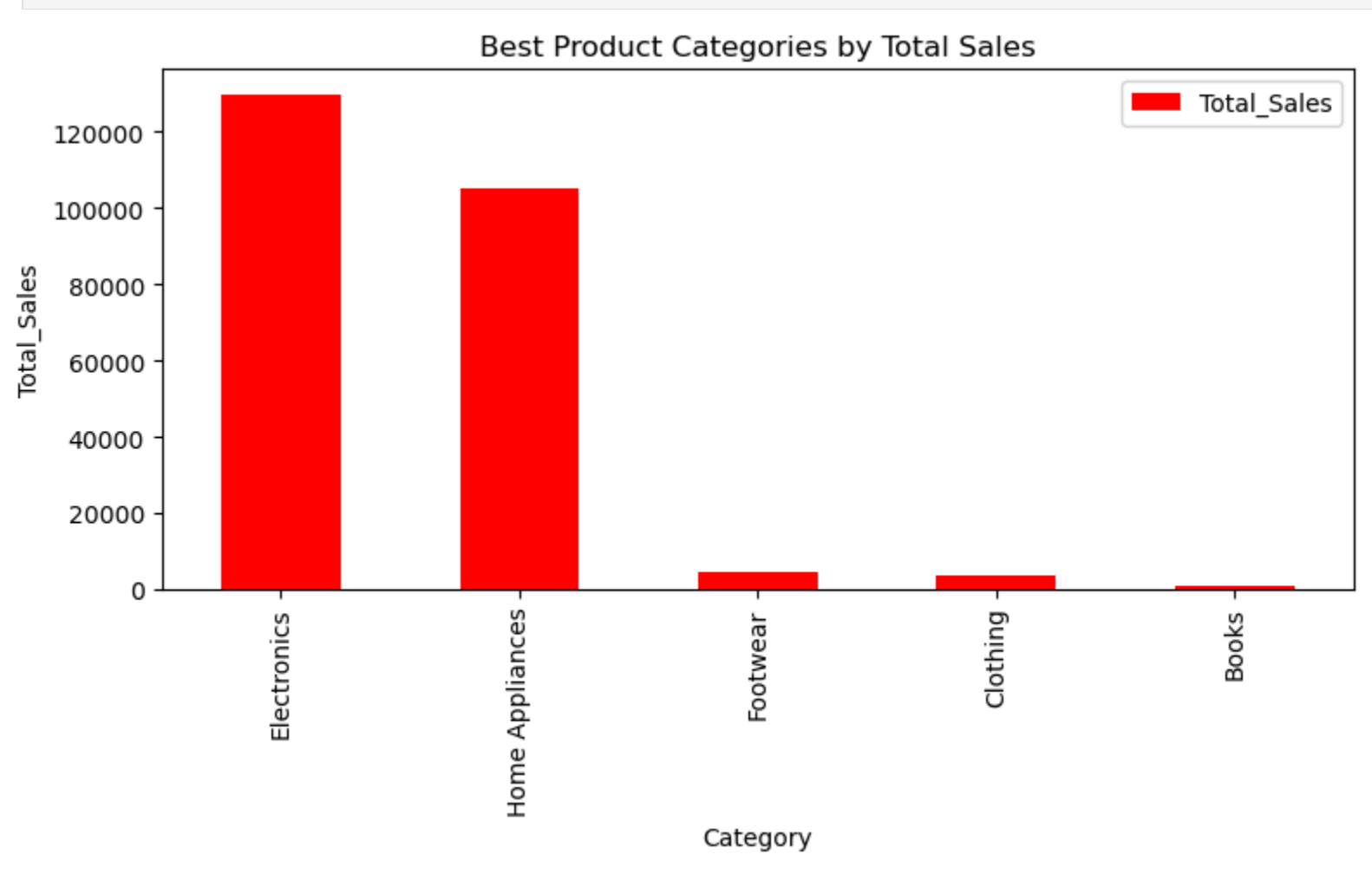
In [44]:
```python
best_product_categories = Amason_sales_data.groupby("Category")[["Total_Sales"]].sum().sort_values(by = "Total_Sales",ascending=False)
```

In [45]:
```python
best_product_categories
```

Out[45]:

Category	Total_Sales
Electronics	129950
Home Appliances	105000
Footwear	4320
Clothing	3540
Books	1035

In [47]:
```python
best_product_categories.plot(kind = "bar",color ="Red",figsize=(8,5))
plt.xlabel("Category")
plt.ylabel("Total_Sales")
plt.title("Best Product Categories by Total Sales")
plt.tight_layout()
plt.show()
```



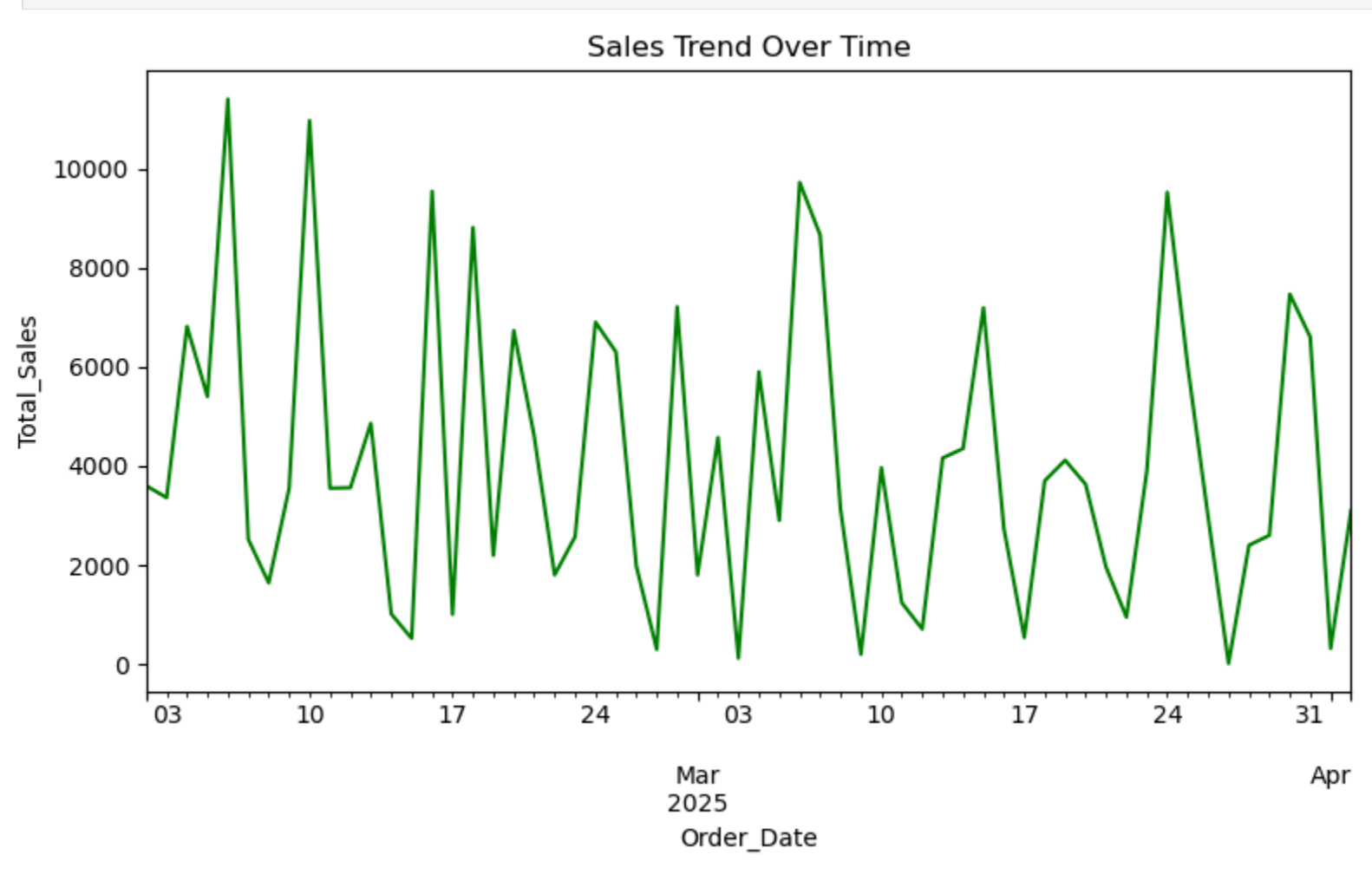
In [48]:
```python
##  Sales Trend Over Time
```

In [49]:
```python
daily_sales = Amason_sales_data.groupby("Order_Date")["Total_Sales"].sum().sort_index()
```

In [50]:
```python
daily_sales
```

Out[50]:
```
Order_Date
2025-02-02     3600
2025-02-03     3360
2025-02-04     6815
2025-02-05     5400
2025-02-06    11400
2025-02-07     2520
2025-02-08     1640
2025-02-09     3550
2025-02-10    10965
2025-02-11     3550
2025-02-12     3560
2025-02-13     4860
2025-02-14     1015
2025-02-15      520
2025-02-16     9540
2025-02-17     1005
2025-02-18     8810
2025-02-19     2195
2025-02-20     6730
2025-02-21     4600
2025-02-22     1800
2025-02-23     2570
2025-02-24     6900
2025-02-25     6300
2025-02-26     1980
2025-02-27      300
2025-02-28     7210
2025-03-01     1800
2025-03-02     4575
2025-03-03      120
2025-03-04     5900
2025-03-05     2900
2025-03-06     9720
2025-03-07     8660
2025-03-08     3125
2025-03-09      200
2025-03-10     3965
2025-03-11     1240
2025-03-12      710
2025-03-13     4160
2025-03-14     4350
2025-03-15     7190
2025-03-16     2735
2025-03-17      540
2025-03-18     3700
2025-03-19     4115
2025-03-20     3630
2025-03-21     1960
2025-03-22      950
2025-03-23     3900
2025-03-24     9520
2025-03-25     6015
2025-03-26     2970
2025-03-27       15
2025-03-28     2400
2025-03-29     2600
2025-03-30     7465
2025-03-31     6600
2025-04-01      320
2025-04-02     3100
Name: Total_Sales, dtype: int64
```

In [51]:
```python
daily_sales.plot(kind = "line",color = "Green",figsize= (8,5))
plt.xlabel("Order_Date")
plt.ylabel("Total_Sales")
plt.title("Sales Trend Over Time")
plt.tight_layout()
plt.show()
```



In [52]:
```python
## Sales By Location
```

In [55]:
```python
Amason_sales_data.columns
```

Out[55]:
```
Index(['Order_ID', 'Order_Date', 'Product', 'Category', 'Price', 'Quantity',
       'Total_Sales', 'Customer_Name', 'Customer_Location', 'Payment_Method',
       'Status'],
      dtype='object')
```

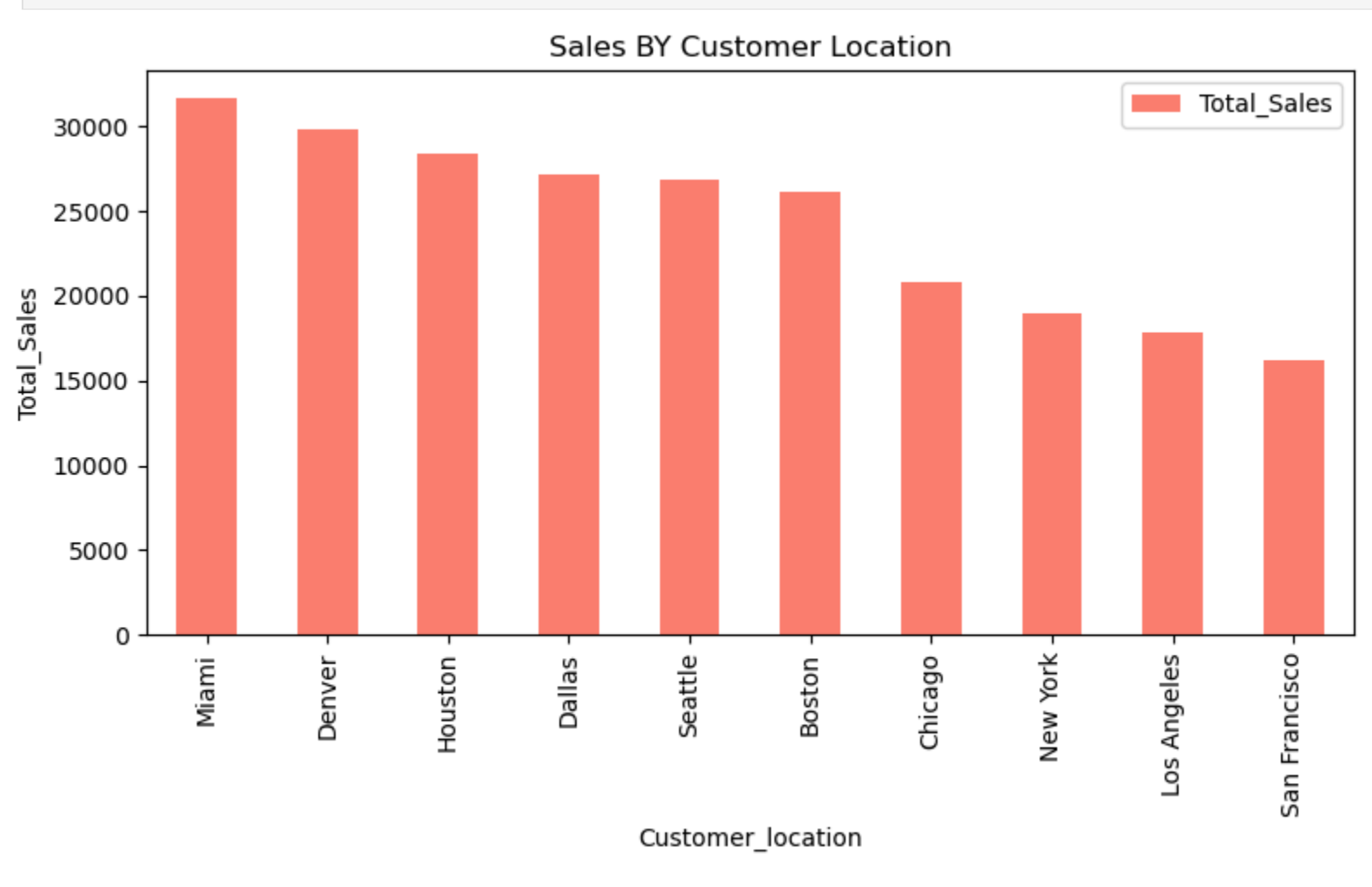
In [73]:
```python
sales_location =  Amason_sales_data.groupby("Customer_Location")[["Total_Sales"]].sum().sort_values("Total_Sales",ascending=False)
```

In [75]:
```python
sales_location
```

Out[75]:

Customer_Location	Total_Sales
Miami	31700
Denver	29785
Houston	28390
Dallas	27145
Seattle	26890
Boston	26170
Chicago	20810
New York	18940
Los Angeles	17820
San Francisco	16195

In [101…
```python
sales_location.plot(kind = "bar",color = "salmon",figsize=(8,5))
plt.xlabel("Customer_location")
plt.title("Sales BY Customer Location")
plt.ylabel("Total_Sales")
plt.tight_layout()
plt.show()
```



In [ ]:
```python
##Order Status Distribution
```

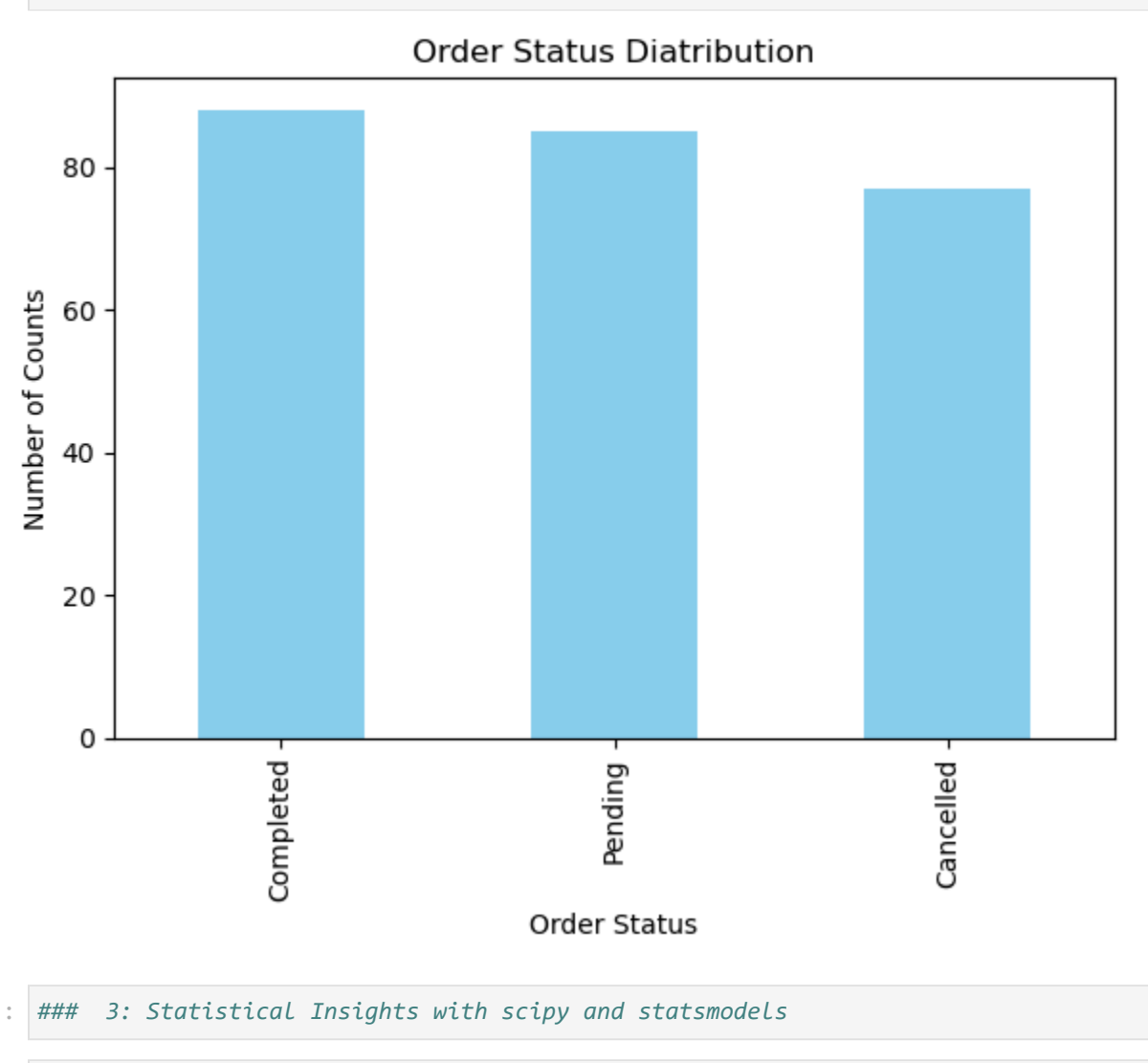
In [103…
```python
Amason_sales_data.columns
```

Out[103…
```
Index(['Order_ID', 'Order_Date', 'Product', 'Category', 'Price', 'Quantity',
       'Total_Sales', 'Customer_Name', 'Customer_Location', 'Payment_Method',
       'Status'],
      dtype='object')
```

In [105…
```python
Amason_sales_data["Status"].value_counts()
```

Out[105…
```
Status
Completed    88
Pending      85
Cancelled    77
Name: count, dtype: int64
```

In [119…
```python
Amason_sales_data["Status"].value_counts().plot(kind = "bar",figsize=(6,5),color ="skyblue")
plt.xlabel("Order Status")
plt.ylabel("Number of Counts")
plt.title("Order Status Diatribution")
plt.tight_layout()
plt.show()
```



In [ ]:
```python
###  3: Statistical Insights with scipy and statsmodels
```

In [246…
```python
import pandas as pd
```

In [248…
```python
import seaborn as sns
```

In [250…
```python
import matplotlib.pyplot as plt
```

In [ ]:
```python
### correlation Matrix
```

In [252…
```python
cor = Amason_sales_data[["Price", "Quantity", "Total_Sales"]].corr()
```

In [254…
```python
cor
```

Out[254…

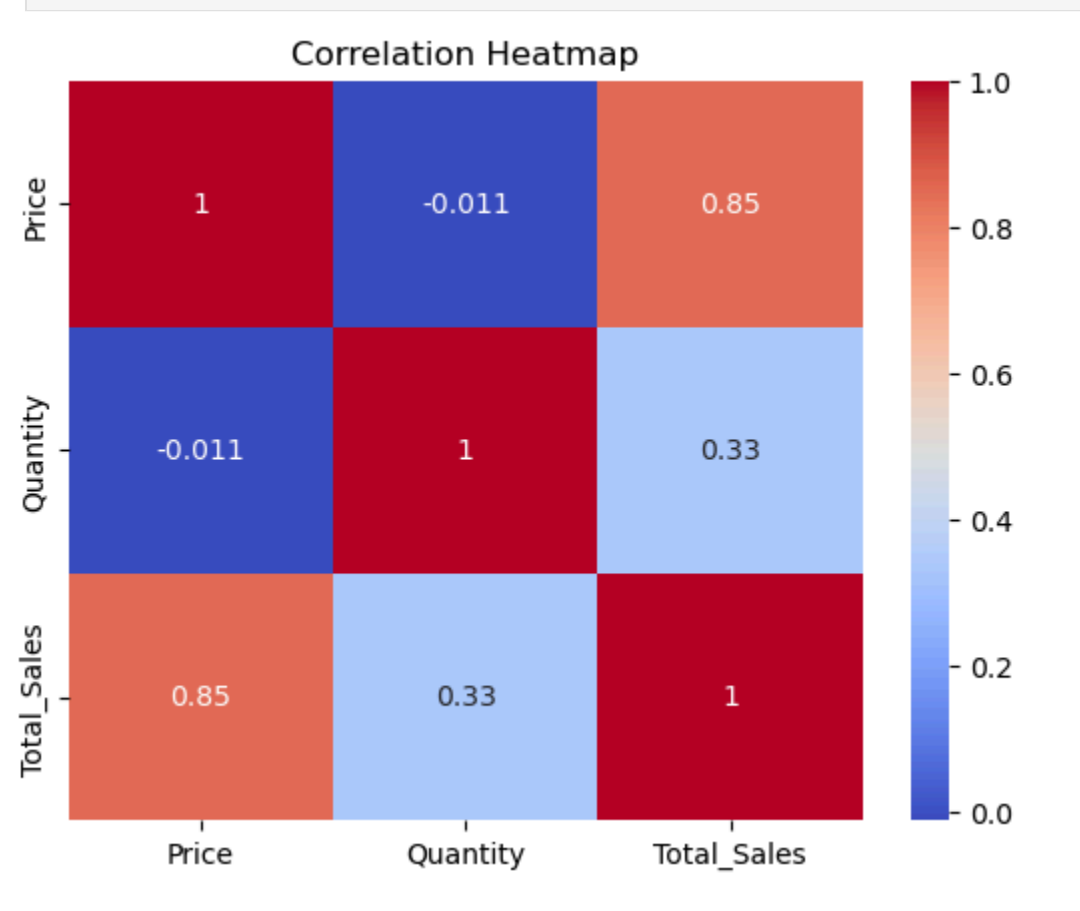
	Price	Quantity	Total_Sales
Price	1.000000	-0.010858	0.846673
Quantity	-0.010858	1.000000	0.332444
Total_Sales	0.846673	0.332444	1.000000

+1 = Perfect Positive Correlation,As price increases, total sales also tend to increas. logical, because Total_Sales = Price × Quantity 0 = No Correlation –1 = Perfect Negative Correlation Price and Quantity aren't related A product's price doesn't influence how many are bought in this dataset

In [ ]:
```python
#In this data, Total Sales depends more on Price than Quantity, and Price doesn’t affect Quantity bought.
```

In [ ]:
```python
# by heat map we can see clearly
```

In [256…
```python
sns.heatmap(cor,annot=True,cmap="coolwarm") #  annot=True   This means annotate the cells — display the numeric values (correlation coefficients) inside each box.
plt.title("Correlation Heatmap")
plt.show()
```



In [ ]:
```python
# ANOVA: Are Total Sales significantly different between categories?
```

In [258…
```python
from scipy import stats
```

In [ ]:
```python
# Grouping sales by category
```

In [260…
```python
Amason_sales_data.columns
```

Out[260…
```
Index(['Order_ID', 'Order_Date', 'Product', 'Category', 'Price', 'Quantity',
       'Total_Sales', 'Customer_Name', 'Customer_Location', 'Payment_Method',
       'Status'],
      dtype='object')
```

In [264…
```python
sales_category = [group["Total_Sales"].values for name,group in Amason_sales_data.groupby("Category")]
```

In [266…
```python
sales_category
```

Out[266…
```
[array([30, 15, 75, 15, 15, 15, 75, 45, 15, 75, 75, 75, 45, 15, 60, 15, 15,
        15, 75, 75, 15, 75, 30, 60, 15], dtype=int64),
 array([ 20,  60, 160,  20,  20, 100,  80,  40, 100, 120,  80, 100, 200,
         40, 200,  40,  60, 120, 200,  20,  60, 200,  80,  20,  80,  40,
        200,  40, 100,  20, 160, 160, 200,  80,  80,  80,  40,  40,  40,
         40], dtype=int64),
 array([ 400,  450,  600,  500,  500, 1600,  600, 1000,  400,  300,  100,
         500,  500, 2000, 2400,  300, 2400, 1600,  150,  300,  150, 2000,
        2400,  450,  300,  500,  500,  300, 4000, 2500,  300,  300,  100,
         750, 2400, 2500,  200,  200,  300,  300, 1500, 2500,  750,  750,
        3200, 1000,  600,  450,  300, 1000,  200, 2500, 3200, 3200,  500,
        1000, 4000,  500, 2400,  500,  300,  750, 1000,  800, 2500,  450,
         750, 1600,  300, 1000, 1500,  300,  300,  400,  150,  750,  200,
         600, 1500, 2400,  100, 1000,  750, 4000,  800,  300,  300, 1500,
        2400, 2400,  300, 1500,  500, 3200, 1500, 1000,  200, 2400,  500,
         800,  750,  400,  300,  450,  500,  400, 4000, 2500,  300,  100,
         750,  150, 2000, 2000, 2000,  800,  450, 2000], dtype=int64),
 array([180, 120, 180, 180, 120, 240, 120, 240, 300, 300,  60,  60, 180,
        300, 180, 120, 120,  60, 120, 300,  60, 120, 240,  60, 180,  60,
        120], dtype=int64),
 array([1800, 1200, 4800, 3600,  600, 1800, 2400, 3600, 1200, 4800, 4800,
        4800,  600, 2400, 2400, 6000, 3000, 1200, 6000, 2400, 2400, 2400,
        4800, 1800, 2400, 2400, 2400, 1200, 3600,  600, 2400, 3600, 3000,
         600, 3600, 1200, 1200, 1200, 3600, 1200], dtype=int64)]
```

In [270…
```python
f_stat,p_val = stats.f_oneway(*sales_category)      # Or anova_result = stats.f_oneway(*category_groups)
                                                    # print("ANOVA F-statistic:", anova_result.statistic)
                                                    # print("ANOVA p-value:", anova_result.pvalue)
```

In [276…
```python
print("f_statistics:",f_stat)
print("p_value:",p_val)
```

```
f_statistics: 53.463921351737696
p_value: 2.4079237572585064e-32
```

In [294…
```python
if p_val<0.05:
    print("Significant difference found between categories.")
else:
    print("no Significant difference found between categories.")
```

```
Significant difference found between categories.
```

In [317…
```python
import statsmodels.api as sm
```

In [319…
```python
 # 3. Linear Regression: Predict Total Sales using Price & Quantity
```

In [ ]:
```python
#Predicting sales from price and quantity linear Regression is useful
```

In [321…
```python
X = Amason_sales_data[["Price","Quantity"]]
```

In [323…
```python
Y = Amason_sales_data["Total_Sales"]
```

In [327…
```python
X = sm.add_constant(X) ## Add Constant(intercept)
```

In [329…
```python
model = sm.OLS(Y,X).fit()
```

In [331…
```python
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Total_Sales   R-squared:                       0.834
Model:                            OLS   Adj. R-squared:                  0.832
Method:                 Least Squares   F-statistic:                     618.6
Date:                Fri, 30 May 2025   Prob (F-statistic):           6.56e-97
Time:                        21:43:36   Log-Likelihood:                -1913.2
No. Observations:                 250   AIC:                             3832.
Df Residuals:                     247   BIC:                             3843.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -840.4835     78.556    -10.699      0.000    -995.208    -685.759
Price          2.7974      0.085     32.760      0.000       2.629       2.966
Quantity     299.2809     22.737     13.163      0.000     254.498     344.064
==============================================================================
Omnibus:                       17.219   Durbin-Watson:                   2.259
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.569
Skew:                           0.018   Prob(JB):                     2.33e-12
Kurtosis:                       5.267   Cond. No.                     1.28e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.28e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:
```python
#I used linear regression to predict Total Sales based on Price and Quantity.
#The model had high accuracy (R² = 0.834),
# showing that these two features strongly influence sales.
# I used statsmodels in Python to find this and both variables were statistically significant.
```