programming-project

August 22, 2024

1 Programming for Analytics: Group Project DT- B1

1.1 Basic Information of the Dataset

[107]:

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

# Load the dataset
data = pd.read_csv('/Users/ashwin/Documents/NMIMS Trimester 1/Programming for␣
↪Analytics/Python/ecommerce_sales_analysis.csv')
data
```

[107]:

```
     product_id  product_name       category   price  review_score  \
0             1     Product_1       Clothing  190.40           1.7
1             2     Product_2  Home & Kitchen  475.60           3.2
2             3     Product_3           Toys  367.34           4.5
3             4     Product_4           Toys  301.34           3.9
4             5     Product_5          Books   82.23           4.2
..          …           …             …      …            …
995         996   Product_996  Home & Kitchen   50.33           3.6
996         997   Product_997  Home & Kitchen  459.07           4.8
997         998   Product_998         Sports   72.73           1.3
998         999   Product_999         Sports  475.37           1.2
999        1000  Product_1000           Toys  225.77           2.1

     review_count  sales_month_1  sales_month_2  sales_month_3  sales_month_4  \
0             220            479            449             92            784
1             903             21            989            861            863
2             163            348            558            567            143
3             951            725            678             59             15
4             220            682            451            649            301
..            …            …            …            …            …
995           494            488            359            137            787
996           701             18            906            129             78
997           287            725            109            193            657
998           720            196            191            315            622
999           114            890            903            983            769
```

```
     sales_month_5  sales_month_6  sales_month_7  sales_month_8  \
0              604            904            446            603
1              524            128            610            436
2              771            409            290            828
3              937            421            670            933
4              620            293            411            258
..             …              …              …              …
995            678            970            282            155
996             19            110            403            683
997            215            337            664            476
998            854            122             65            938
999            134            704            648            400

     sales_month_9  sales_month_10  sales_month_11  sales_month_12
0              807             252             695             306
1              176             294             772             353
2              340             667             267             392
3               56             157             168             203
4              854             548             770             257
..             …               …               …               …
995             57             575             634             393
996            104             858             729             474
997            265             344             888             654
998            521             268              60             394
999            495             839             611             110

[1000 rows x 18 columns]
```

[108]:
```
data.info
```

[108]:
```
<bound method DataFrame.info of      product_id  product_name       category
price  review_score  \
0             1     Product_1       Clothing  190.40           1.7
1             2     Product_2  Home & Kitchen  475.60           3.2
2             3     Product_3           Toys  367.34           4.5
3             4     Product_4           Toys  301.34           3.9
4             5     Product_5          Books   82.23           4.2
..          …            …              …      …             …
995         996   Product_996  Home & Kitchen   50.33           3.6
996         997   Product_997  Home & Kitchen  459.07           4.8
997         998   Product_998         Sports   72.73           1.3
998         999   Product_999         Sports  475.37           1.2
999        1000  Product_1000           Toys  225.77           2.1

     review_count  sales_month_1  sales_month_2  sales_month_3  sales_month_4  \
0             220            479            449             92            784
1             903             21            989            861            863
```

```
2              163          348          558          567          143
3              951          725          678           59           15
4              220          682          451          649          301
..             ...          ...          ...          ...          ...
995            494          488          359          137          787
996            701           18          906          129           78
997            287          725          109          193          657
998            720          196          191          315          622
999            114          890          903          983          769

     sales_month_5  sales_month_6  sales_month_7  sales_month_8  \
0              604            904            446            603
1              524            128            610            436
2              771            409            290            828
3              937            421            670            933
4              620            293            411            258
..             ...            ...            ...            ...
995            678            970            282            155
996             19            110            403            683
997            215            337            664            476
998            854            122             65            938
999            134            704            648            400

     sales_month_9  sales_month_10  sales_month_11  sales_month_12
0              807             252             695             306
1              176             294             772             353
2              340             667             267             392
3               56             157             168             203
4              854             548             770             257
..             ...             ...             ...             ...
995             57             575             634             393
996            104             858             729             474
997            265             344             888             654
998            521             268              60             394
999            495             839             611             110

[1000 rows x 18 columns]>
```

[109]:
```python
data.describe()
```

[109]:
```
        product_id        price  review_score  review_count  sales_month_1  \
count  1000.000000  1000.000000   1000.000000   1000.000000    1000.000000
mean    500.500000   247.677130      3.027600    526.506000     498.306000
std     288.819436   144.607983      1.171243    282.269932     289.941478
min       1.000000     7.290000      1.000000      1.000000       0.000000
25%     250.750000   121.810000      2.000000    283.750000     245.500000
50%     500.500000   250.920000      3.100000    543.000000     507.500000
```

```
75%      750.250000   373.435000      4.000000    772.000000     740.750000
max     1000.000000   499.860000      5.000000    999.000000    1000.000000

       sales_month_2  sales_month_3  sales_month_4  sales_month_5  \
count    1000.000000    1000.000000    1000.000000    1000.000000
mean      507.661000     506.739000     503.823000     487.194000
std       285.992689     294.010873     286.645567     287.844324
min         2.000000       0.000000       0.000000       0.000000
25%       262.500000     243.750000     261.500000     221.000000
50%       508.000000     493.000000     501.500000     497.000000
75%       756.250000     777.250000     749.500000     727.000000
max      1000.000000     999.000000    1000.000000    1000.000000

       sales_month_6  sales_month_7  sales_month_8  sales_month_9  \
count    1000.000000    1000.000000    1000.000000    1000.000000
mean      491.653000     507.011000     504.569000     491.934000
std       289.234018     291.047287     289.945691     287.514731
min         0.000000       0.000000       5.000000       0.000000
25%       236.000000     254.000000     240.500000     247.250000
50%       479.500000     522.500000     499.500000     495.500000
75%       740.500000     757.250000     762.250000     735.250000
max      1000.000000    1000.000000    1000.000000    1000.000000

       sales_month_10  sales_month_11  sales_month_12
count     1000.000000      1000.00000     1000.000000
mean       514.798000       505.83800      500.386000
std        288.710119       288.82451      278.509459
min          1.000000         0.00000        4.000000
25%        267.000000       251.25000      259.000000
50%        532.000000       502.00000      500.500000
75%        770.250000       761.00000      730.000000
max       1000.000000      1000.00000     1000.000000
```

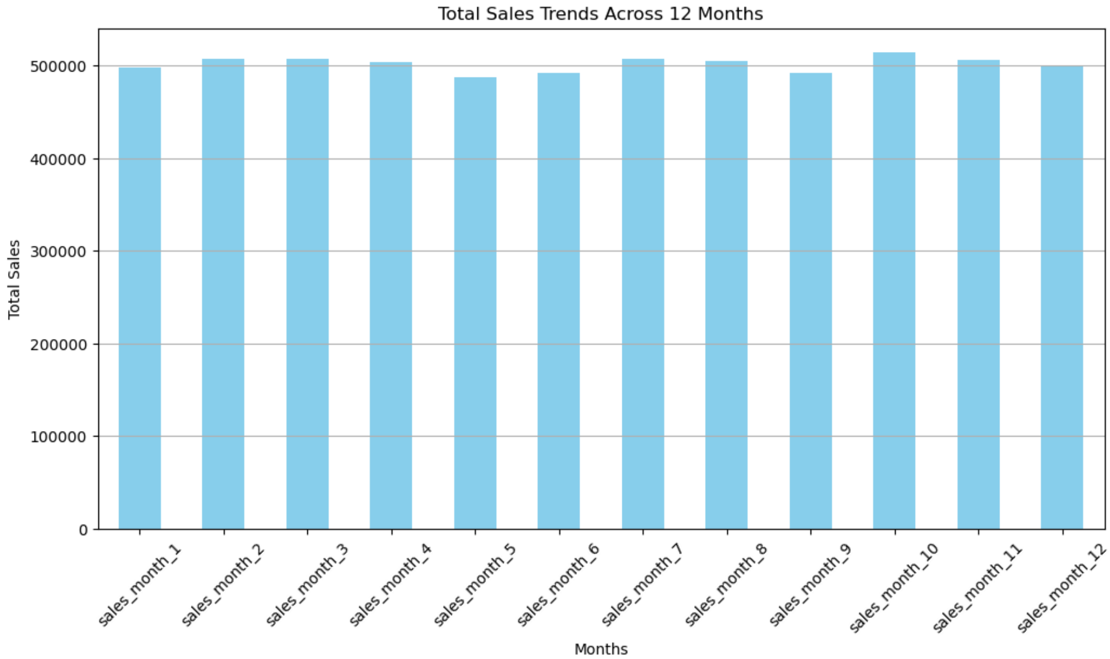
1.1.1 1. Sales Trends Across 12 Months

Objective

Analyze the sales trends over the 12-month period to identify patterns and fluctuations.

[5]:
```
# Calculate total sales per month
monthly_sales = data.loc[:, 'sales_month_1':'sales_month_12'].sum()

# Plotting the sales trends
plt.figure(figsize=(12, 6))
monthly_sales.plot(kind='bar', color='skyblue')
plt.title('Total Sales Trends Across 12 Months')
plt.xlabel('Months')
plt.ylabel('Total Sales')
```

```
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



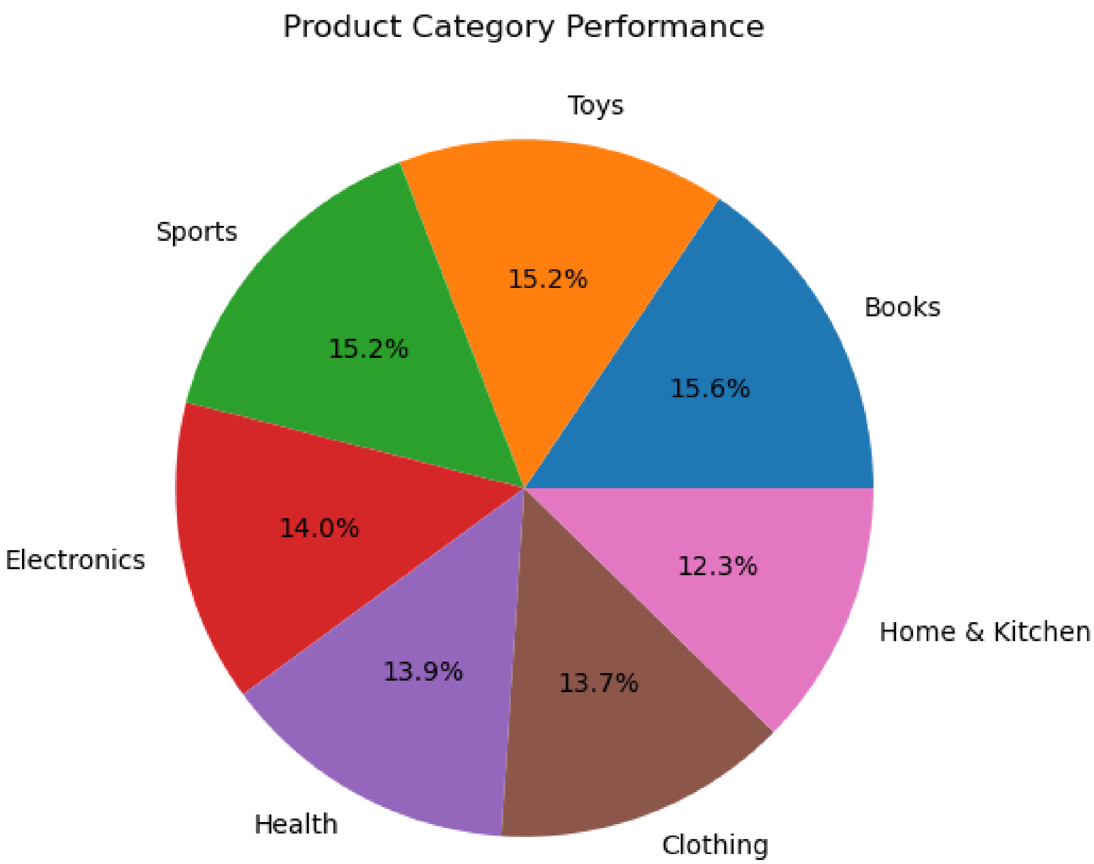
Inference: The above graph shows the relation between Total Sales across Months.

1.1.2 2. Product Performance

Objective

Evaluate the popularity and sales of product categories and items.

[110]:
```
# Calculate total sales per product
data['total_sales'] = data.loc[:, 'sales_month_1':'sales_month_12'].sum(axis=1)

# Group by category to find total sales per category
category_performance = data.groupby('category')['total_sales'].sum().
 ↪sort_values(ascending=False)

# Plotting product category performance
plt.figure(figsize=(12, 6))
category_performance.plot(kind='pie', autopct='%0.1f%%', startangle=0)
plt.title('Product Category Performance')
plt.ylabel('')
plt.show()
```



1.1.3 Inference:

Here we can infer that most popular item sold is Books

1.1.4 3. Revenue Analysis

Objective

Assess contributions from different segments and identify high-value products.

[111]:
```
# Calculate revenue per product
data['revenue'] = data['total_sales'] * data['price']

# Identify high-value products
high_value_products = data.sort_values(by='revenue', ascending=False).head(10)

# Display high-value products
print(high_value_products[['product_name', 'category', 'revenue']])
```

```
    product_name     category     revenue
305  Product_306        Books  3812158.55
531  Product_532        Books  3763945.80
52    Product_53       Sports  3722796.00
228  Product_229  Electronics  3698409.92
390  Product_391        Books  3650361.00
522  Product_523         Toys  3566043.61
140  Product_141        Books  3549917.34
112  Product_113     Clothing  3520168.40
751  Product_752       Health  3519126.48
475  Product_476         Toys  3515858.12
```

1.1.5 Inference:

Output is showing largest 10 values of revenue in descending order.

1.1.6 4. Customer Retention

Objective

Analyze purchase behavior to develop retention strategies.

[112]:
```python
# Calculate average monthly sales per product
data['average_monthly_sales'] = data.loc[:, 'sales_month_1':'sales_month_12'].
    ↪mean(axis=1)

# Identify products with consistent sales
retention_products = data[data['average_monthly_sales'] >␣
    ↪data['average_monthly_sales'].mean()]

# Display retention products
print(retention_products[['product_name', 'average_monthly_sales']])
```

```
     product_name  average_monthly_sales
0       Product_1             535.083333
1       Product_2             502.250000
4       Product_5             507.833333
5       Product_6             537.750000
15     Product_16             503.500000
..            …                    …
986   Product_987             593.000000
989   Product_990             639.583333
990   Product_991             552.333333
994   Product_995             560.500000
999  Product_1000             623.833333

[486 rows x 2 columns]
```

1.1.7 Inference:

Shows the average monthly sales of each product

1.2 5. Identifying Top-Selling Products

Objective

Find the top-selling products based on total sales across all months.

[113]:
```
# Calculate total sales per product
data['total_sales'] = data.loc[:, 'sales_month_1':'sales_month_12'].sum(axis=1)

# Sort products by total sales in descending order
top_products = data.sort_values(by='total_sales', ascending=False).head(10)

# Display top-selling products
print(top_products[['product_name', 'category', 'total_sales']])
```

```
    product_name     category  total_sales
223  Product_224  Electronics         9151
285  Product_286     Clothing         8921
733  Product_734       Health         8914
904  Product_905       Sports         8783
179  Product_180       Sports         8775
852  Product_853        Books         8765
238  Product_239       Health         8724
923  Product_924  Electronics         8525
936  Product_937  Electronics         8459
196  Product_197         Toys         8418
```

1.2.1 Inference:

Total sales achieved for each product category.

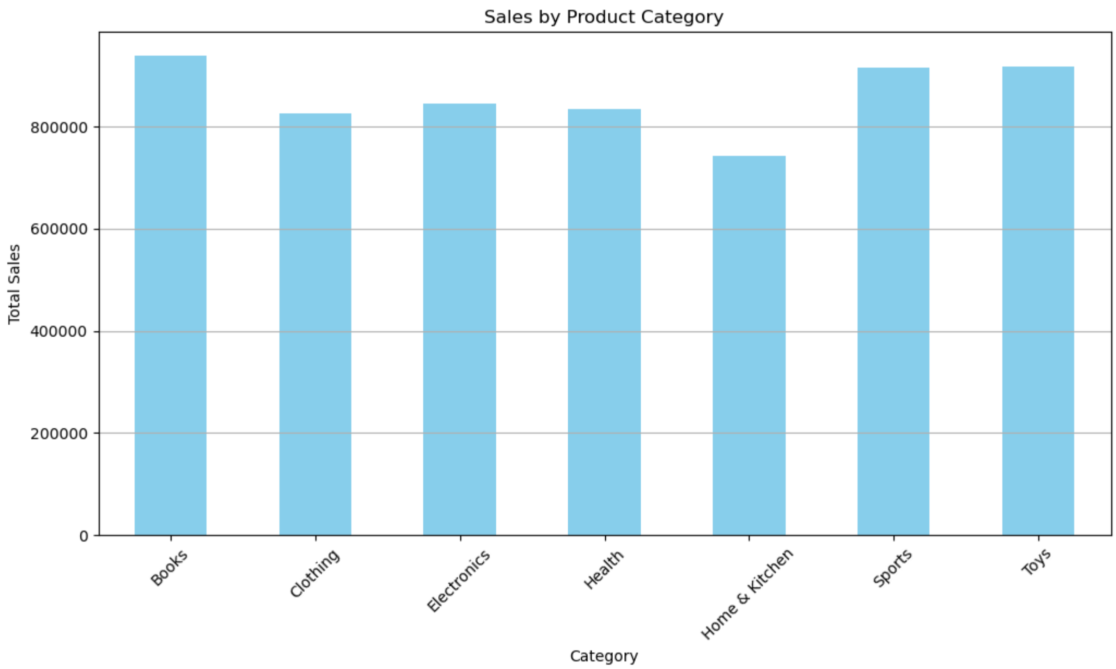
1.3 6. Analyzing Sales by Category

Objective

Investigate sales trends and performance across different product categories.

[114]:
```
# Group by category to find total sales per category
category_sales = data.groupby('category')['total_sales'].sum()

# Plotting sales by category
plt.figure(figsize=(12, 6))
category_sales.plot(kind='bar', color='skyblue')
plt.title('Sales by Product Category')
plt.xlabel('Category')
plt.ylabel('Total Sales')
```

```
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



1.3.1 Inference:

Total sales of books is highest, followed by sports and toys.

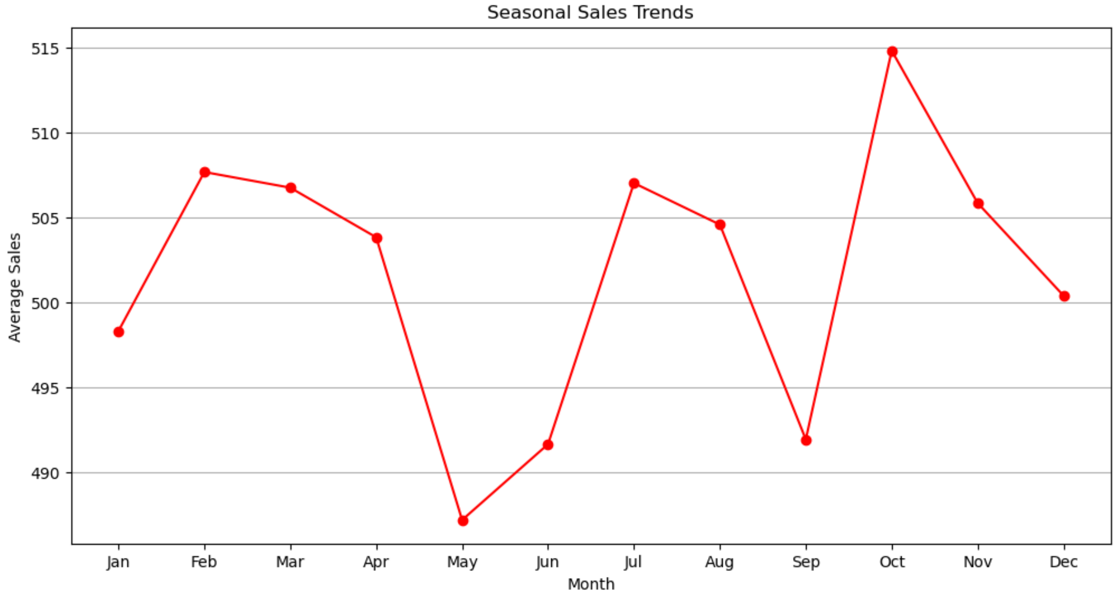
1.4 7. Identifying Seasonal Trends

Objective

Detect any seasonal patterns or fluctuations in sales across the 12-month period.

[115]:
```
# Calculate average sales per month
avg_monthly_sales = data.loc[:, 'sales_month_1':'sales_month_12'].mean(axis=0)

# Plotting seasonal sales trends
plt.figure(figsize=(12, 6))
avg_monthly_sales.plot(kind='line', color='red', marker='o')
plt.title('Seasonal Sales Trends')
plt.xlabel('Month')
plt.ylabel('Average Sales')
plt.xticks(range(12), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',␣
    ↪'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid(axis='y')
```

```
plt.show()
```



1.4.1 Inference:

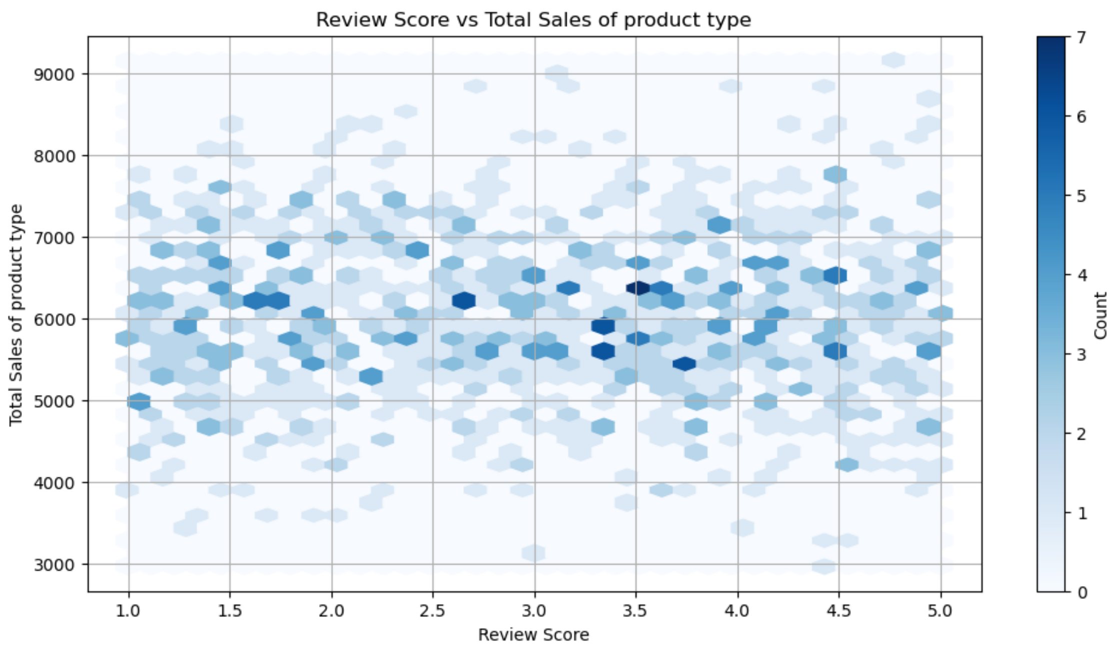
Highest average sales in Ecommerce in the month of October and lowest sales in the month of May.

1.5 8. Analyzing Product Reviews

Objective

Investigate the relationship between product reviews and sales performance.

[116]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# Calculate total sales for each product across all months
data['total_sales'] = data.loc[:, 'sales_month_1':'sales_month_12'].sum(axis=1)

plt.figure(figsize=(12, 6))
plt.hexbin(data['review_score'], data['total_sales'], gridsize=35, cmap='Blues')
plt.colorbar(label='Count')
plt.title('Review Score vs Total Sales of product type')
plt.xlabel('Review Score')
plt.ylabel('Total Sales of product type')
plt.grid(True)
plt.show()
```



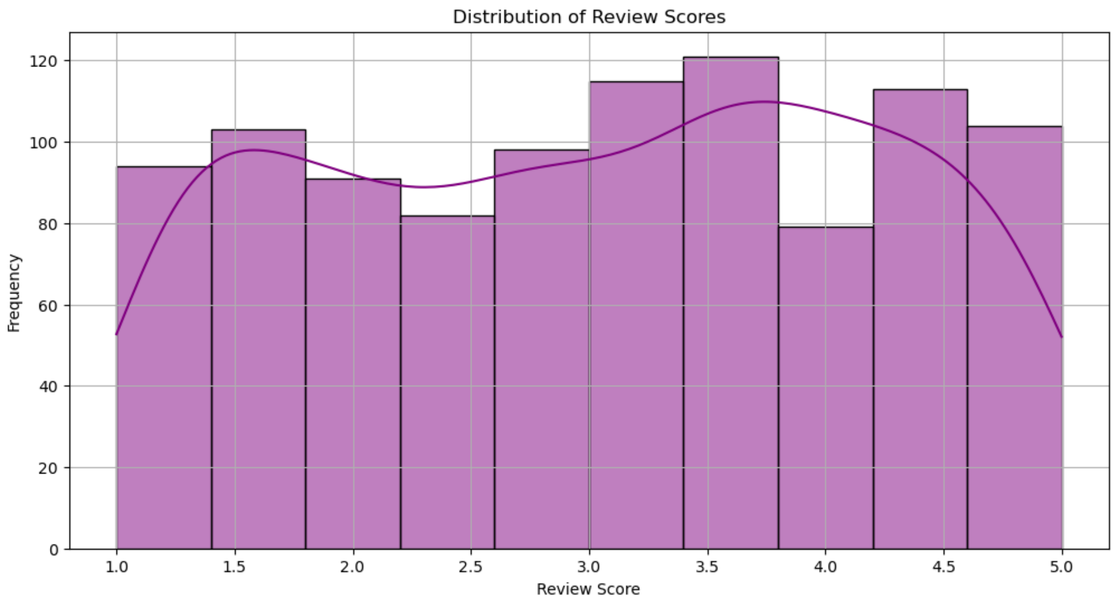
1.5.1 Inference:

- Product Concentration: There seems to be a higher concentration of products with review scores between 3.0 and 4.5, and total sales mostly between 5000 and 7000. This suggests that many products receive moderate to high review scores and have corresponding sales in this range.
- Distribution Across Scores: The hexagons are spread across the review score range, indicating that products with both low and high review scores have varying levels of sales.
- Sales Across Reviews: Products with higher review scores (above 4.0) seem to still vary widely in terms of total sales, suggesting that a good review score doesn't always guarantee high sales, and other factors may also influence sales.

[117]:
```
#Distribution of Review Scores and extension of above graph for clearer␣
↪depiction of data.
plt.figure(figsize=(12, 6))
sns.histplot(data['review_score'], bins=10, kde=True, color='purple')
plt.title('Distribution of Review Scores')
plt.xlabel('Review Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
```

```
with pd.option_context('mode.use_inf_as_na', True):
```



1.5.2 Inference:

From the graph we can infer that frequency of review score is highest around 3.5 to 4 which is around 120 reviews.

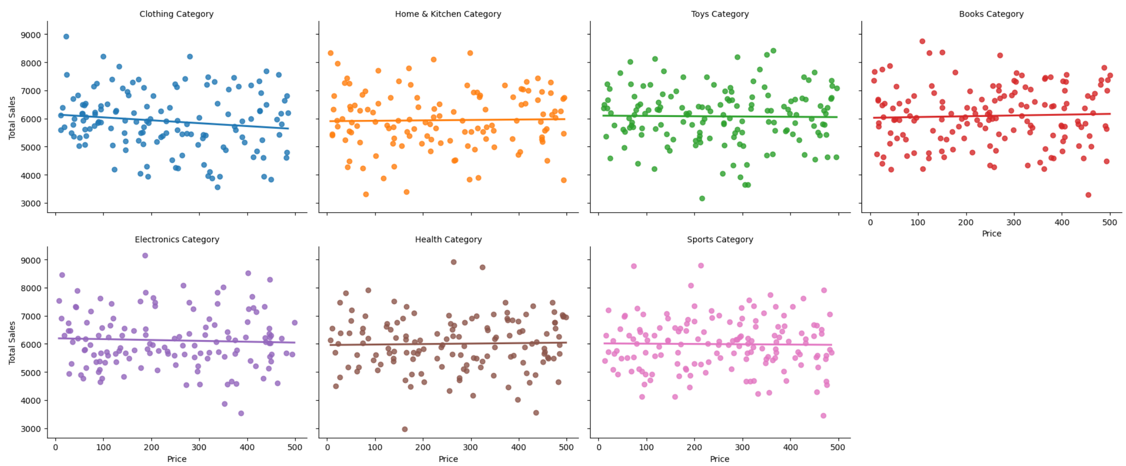
1.6 9. Analysing Product category sales with respect to price.

Objective

Visualize the relationship between two continuous variables, such as price and total sales.

[118]:
```
# Create a facet grid with regression lines
g = sns.lmplot(data=data, x='price', y='total_sales', hue='category',␣
↪col='category', col_wrap=4, height=4, aspect=1.2, ci=None)

# Adjust labels and titles
g.set_axis_labels('Price', 'Total Sales')
g.set_titles(col_template="{col_name} Category")

# Show the plot
plt.show()

#only to know if the slopes are positive or negative to analyse the trends as␣
↪the lines are almost flat.
slopes = []
```

```python
# Iterate over each category
for category in data['category'].unique():
    subset = data[data['category'] == category]
    x = subset['price']
    y = subset['total_sales']
    coeffs = np.polyfit(x, y, 1)  # Fit a linear polynomial (degree 1)
    slope = coeffs[0]  # Slope of the regression line
    slopes.append({'category': category, 'slope': slope})

# Convert slopes to DataFrame for easier handling
slopes_df = pd.DataFrame(slopes)
print(slopes_df)
```




```
      category     slope
0     Clothing -1.032654
1  Home & Kitchen  0.146724
2         Toys -0.105083
3        Books  0.279662
4  Electronics -0.306549
5       Health  0.170300
6       Sports -0.105557
```

1.6.1 Inference:

- If the line has a positive slope, it suggests that higher prices are generally associated with higher total sales.
- If the line has a negative slope, it suggests that higher prices are associated with lower total sales.

So here we can say that for Categories: Electronics, Clothing, Toys and Sports as the price increases the sales decreases. And for other categories the price increase results in higher sales

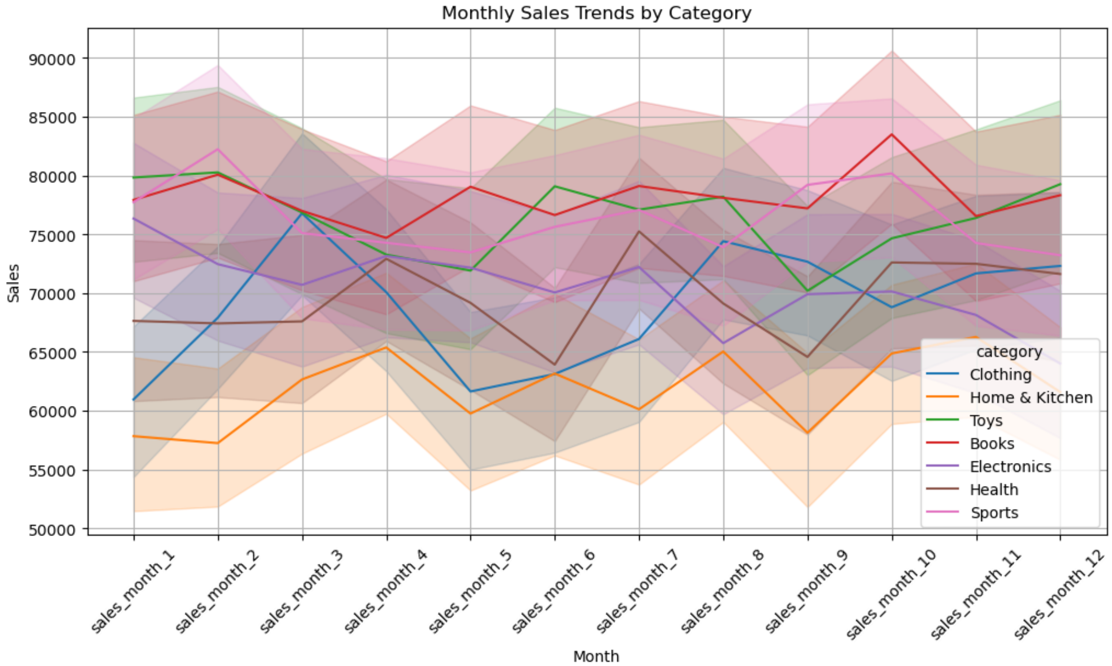
1.7 10. Product wise monthly trends

Objective

Show trends over time, such as monthly sales for a specific product category.

[119]:
```
# Melt the data for easier plotting
monthly_data = data.melt(id_vars=['product_name', 'category'],
                          value_vars=['sales_month_1', 'sales_month_2',␣
 ↪'sales_month_3',
                                      'sales_month_4', 'sales_month_5',␣
 ↪'sales_month_6',
                                      'sales_month_7', 'sales_month_8',␣
 ↪'sales_month_9',
                                      'sales_month_10', 'sales_month_11',␣
 ↪'sales_month_12'],
                          var_name='month', value_name='sales')

# Line plot of monthly sales for each category
plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_data, x='month', y='sales', hue='category',␣
 ↪estimator='sum')
plt.title('Monthly Sales Trends by Category')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



1.7.1 Inference:

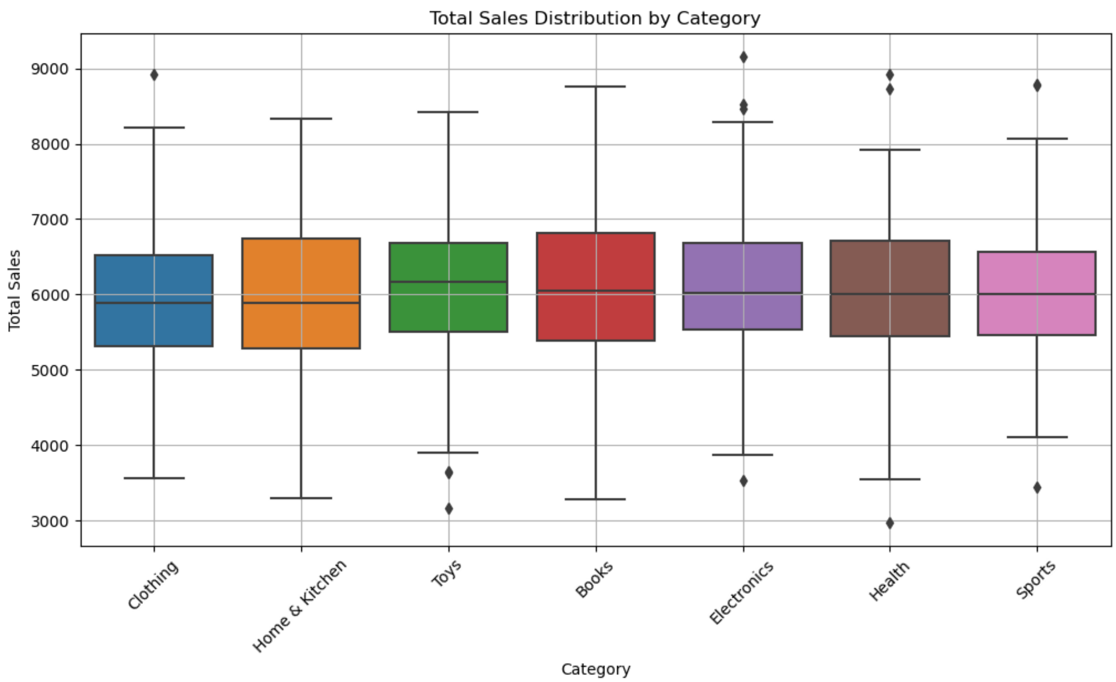
From above graph we can say that highest sales for each category: * Clothing: Sales Month 3 * Home and Kitchen: Sales Month 11 * Toys: Sales Month 1 * Books: Sales Month 10 * Electronics: Sales Month 1 * Health: Sales Month 7 * Sports: Sales Month 2

1.8 11. Determining the sales outliers in each product category

Objective

Analyze the distribution of sales across different categories and identify outliers.

[120]:
```
# Box plot of total sales by category
plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x='category', y='total_sales')
plt.title('Total Sales Distribution by Category')
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid()
plt.show()
```



1.8.1 Inference:

- Clothing: Minimum sales is around 3600 and maximum sales is around 8200.
- Home and Kitchen: Minimum sales is around 3300 and maximum sales is around 8300
- Toys: Minimum sales is around 3900 and maximum sales is around 8400
- Books: Minimum sales is around 3300 and maximum sales is around 8800
- Electronics: Minimum sales is around 3800 and maximum sales is around 8250
- Health: Minimum sales is around 3500 and maximum sales is around 7950
- Sports: Minimum sales is around 4100 and maximum sales is around 8100

Anything beyond this range of maxima and minima are considered to be outliers. Also, box indicates that maximum frequency of sales lies between IQR1 and IQR3, IQR2 being the peak of the sales.

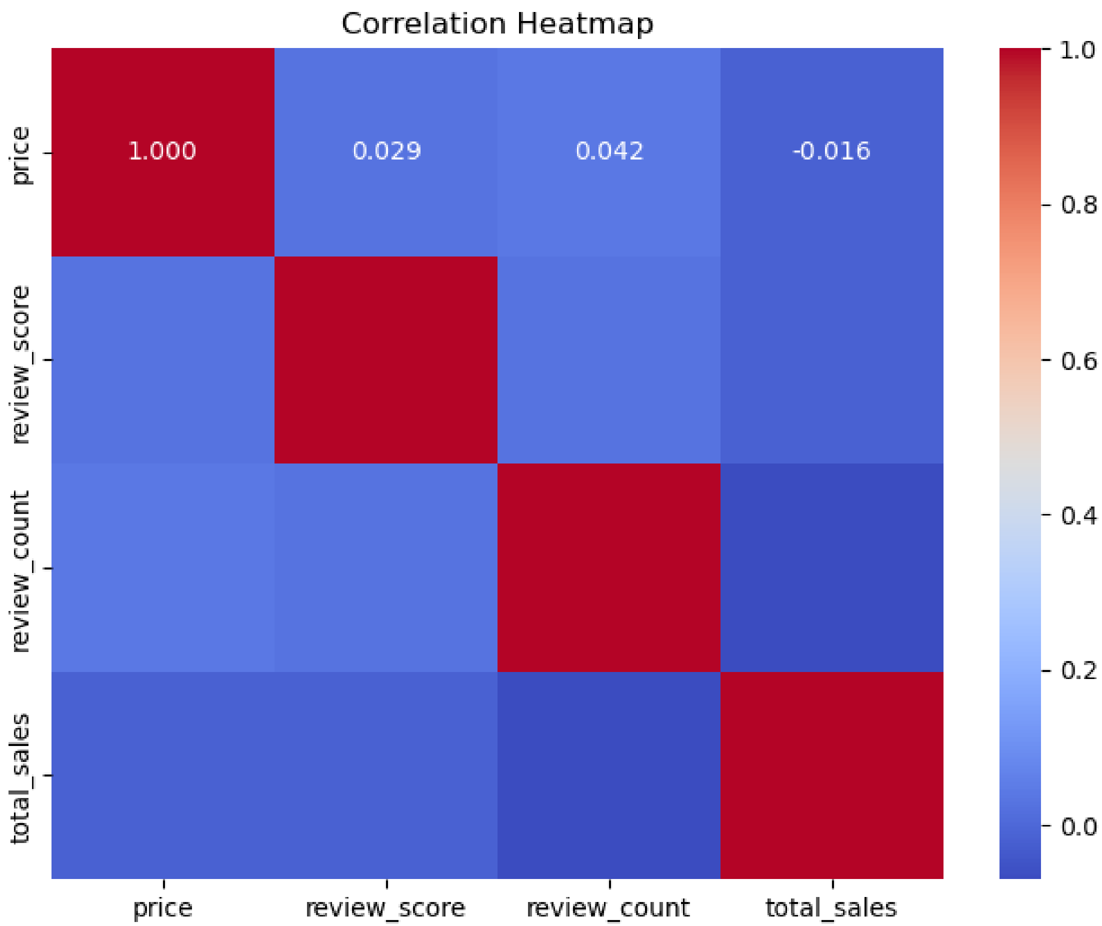
1.9 12. Multivariable Relationship

Objective

Visualize pairwise relationships in the dataset to understand correlations between multiple variables.

[122]:
```
# Calculate the correlation matrix
correlation_matrix = data[['price', 'review_score', 'review_count',␣
↪'total_sales']].corr()

# Heatmap of the correlation matrix
plt.figure(figsize=(8, 6))
```

```python
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.3f')
plt.title('Correlation Heatmap')
plt.show()
```



1.9.1 Inference:

From the visualisation we can infer that there is not much relation between variables with respect to each other. (For example: Change in price doesn't effect the total sales or reviews much).

[ ]: