Project Title - Analyzing Amazon Sales Data

Technologies - Data Science

Domain - E-commerce

In [1]:
```
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
import seaborn as sns
```

In [4]:
```python
df = pd.read_csv(r'C:\Users\Prash\Downloads\Amazon_Sales_data.csv')
df
```

Out[4]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Uni So
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	99
1	Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	28
2	Europe	Russia	Office Supplies	Offline	L	5/2/2014	341417157	5/8/2014	17
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	6/20/2014	514321792	7/5/2014	81
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2/1/2013	115456712	2/6/2013	50
...	...	...	...	...	...	...	...	...	
95	Sub-Saharan Africa	Mali	Clothes	Online	M	7/26/2011	512878119	9/3/2011	8
96	Asia	Malaysia	Fruits	Offline	L	11/11/2011	810711038	12/28/2011	62
97	Sub-Saharan Africa	Sierra Leone	Vegetables	Offline	C	6/1/2016	728815257	6/29/2016	14
98	North America	Mexico	Personal Care	Offline	M	7/30/2015	559427106	8/8/2015	57
99	Sub-Saharan Africa	Mozambique	Household	Offline	L	2/10/2012	665095412	2/15/2012	53

100 rows × 14 columns

In [5]:
```
# Step 2: Data Transformation
# Assuming your date column is named 'order_date'
df['Order Date'] = pd.to_datetime(df['Order Date'])
df
```

Out[5]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Uni Sol
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	669165933	6/27/2010	992
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	963881480	9/15/2012	280
2	Europe	Russia	Office Supplies	Offline	L	2014-05-02	341417157	5/8/2014	177
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	2014-06-20	514321792	7/5/2014	810
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2013-02-01	115456712	2/6/2013	506

In [6]:
```
df['year'] = df['Order Date'].dt.year
df
```

Out[6]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	669165933	6/27/2010	9925
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	963881480	9/15/2012	2804
2	Europe	Russia	Office Supplies	Offline	L	2014-05-02	341417157	5/8/2014	1779
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	2014-06-20	514321792	7/5/2014	8102
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2013-02-01	115456712	2/6/2013	5062
...	...	...	...	...	...	...	...	...	...
95	Sub-Saharan Africa	Mali	Clothes	Online	M	2011-07-26	512878119	9/3/2011	888
96	Asia	Malaysia	Fruits	Offline	L	2011-11-11	810711038	12/28/2011	6267
97	Sub-Saharan Africa	Sierra Leone	Vegetables	Offline	C	2016-06-01	728815257	6/29/2016	1485
98	North America	Mexico	Personal Care	Offline	M	2015-07-30	559427106	8/8/2015	5767
99	Sub-Saharan Africa	Mozambique	Household	Offline	L	2012-02-10	665095412	2/15/2012	5367

100 rows × 15 columns

In [7]:
```
df['month'] = df['Order Date'].dt.month
df
```

Out[7]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	669165933	6/27/2010	9925
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	963881480	9/15/2012	2804
2	Europe	Russia	Office Supplies	Offline	L	2014-05-02	341417157	5/8/2014	1779
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	2014-06-20	514321792	7/5/2014	8102
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2013-02-01	115456712	2/6/2013	5062
...	...	...	...	...	...	...	...	...	...
95	Sub-Saharan Africa	Mali	Clothes	Online	M	2011-07-26	512878119	9/3/2011	888
96	Asia	Malaysia	Fruits	Offline	L	2011-11-11	810711038	12/28/2011	6267
97	Sub-Saharan Africa	Sierra Leone	Vegetables	Offline	C	2016-06-01	728815257	6/29/2016	1485
98	North America	Mexico	Personal Care	Offline	M	2015-07-30	559427106	8/8/2015	5767
99	Sub-Saharan Africa	Mozambique	Household	Offline	L	2012-02-10	665095412	2/15/2012	5367

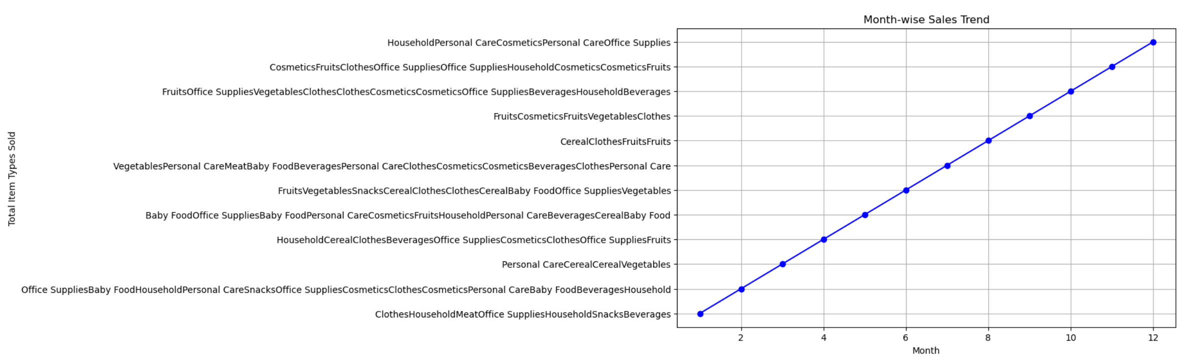
100 rows × 16 columns

In [39]:
```python
# Step 4: Sales Trend Analysis
# Month-wise Sales Trend
monthly_sales = df.groupby('month')[['Item Type']].sum()
monthly_sales
```

Out[39]:

month	Item Type
1	ClothesHouseholdMeatOffice SuppliesHouseholdSn...
2	Office SuppliesBaby FoodHouseholdPersonal Care...
3	Personal CareCerealCerealVegetables
4	HouseholdCerealClothesBeveragesOffice Supplies...
5	Baby FoodOffice SuppliesBaby FoodPersonal Care...
6	FruitsVegetablesSnacksCerealClothesClothesCere...
7	VegetablesPersonal CareMeatBaby FoodBeveragesP...
8	CerealClothesFruitsFruits
9	FruitsCosmeticsFruitsVegetablesClothes
10	FruitsOffice SuppliesVegetablesClothesClothesC...
11	CosmeticsFruitsClothesOffice SuppliesOffice Su...
12	HouseholdPersonal CareCosmeticsPersonal CareOf...

In [47]:
```python
# Assuming 'monthly_sales' DataFrame is already calculated
# If not, use the provided 'groupby' operation to create it

# Plotting the month-wise sales trend
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales['Item Type'], marker='o', linesty
plt.title('Month-wise Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Item Types Sold')
plt.grid(True)
plt.show()
```



Sales Trend Analysis:

Month-wise Sales Trend:

The month-wise sales trend analysis provides insights into the variations in sales throughout the year. Peaks and troughs in sales can be observed, allowing for better understanding of seasonal patterns.

Year-wise Sales Trend of Item

In [38]:
```
yearly_sales = df.groupby('year')[['Item Type']].sum()
yearly_sales
```

Out[38]:

year	Item Type
2010	Baby FoodHouseholdFruitsCosmeticsPersonal Care...
2011	HouseholdVegetablesFruitsOffice SuppliesOffice...
2012	CerealVegetablesClothesMeatHouseholdCosmeticsO...
2013	Office SuppliesPersonal CareFruitsCerealFruits...
2014	Office SuppliesFruitsCerealClothesPersonal Car...
2015	Baby FoodPersonal CareBeveragesBaby FoodOffice...
2016	CosmeticsSnacksPersonal CareCerealCosmeticsClo...
2017	ClothesHouseholdCosmeticsSnacksPersonal CareMe...

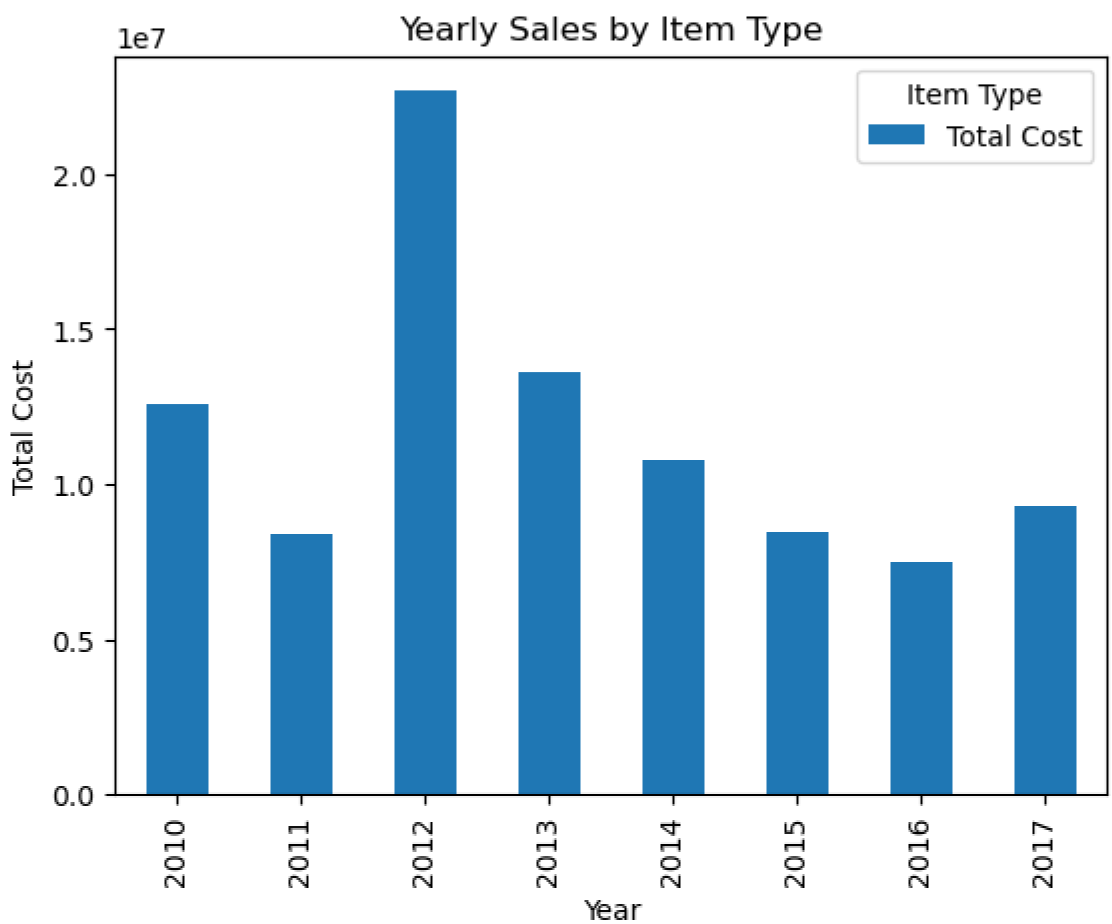
In [46]:
```
yearly_sales = df.groupby('year')[['Item Type','Total Cost']].sum()
yearly_sales
```

Out[46]:

year	Total Cost
2010	12556457.49
2011	8388157.84
2012	22685634.40
2013	13615028.62
2014	10750752.75
2015	8431443.42
2016	7469029.21
2017	9284066.18

In [48]:
```
# Assuming 'yearly_sales' DataFrame is already calculated
# If not, use the provided 'groupby' operation to create it

# Plotting the yearly sales by item type
plt.figure(figsize=(12, 8))
yearly_sales.plot(kind='bar', stacked=True)
plt.title('Yearly Sales by Item Type')
plt.xlabel('Year')
plt.ylabel('Total Cost')
plt.legend(title='Item Type')
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



Year-wise Sales Trend:

Analyzing the year-wise sales trend gives an overview of the overall performance of the business in terms of total cost. This information is crucial for understanding the growth or contraction of sales over different years.

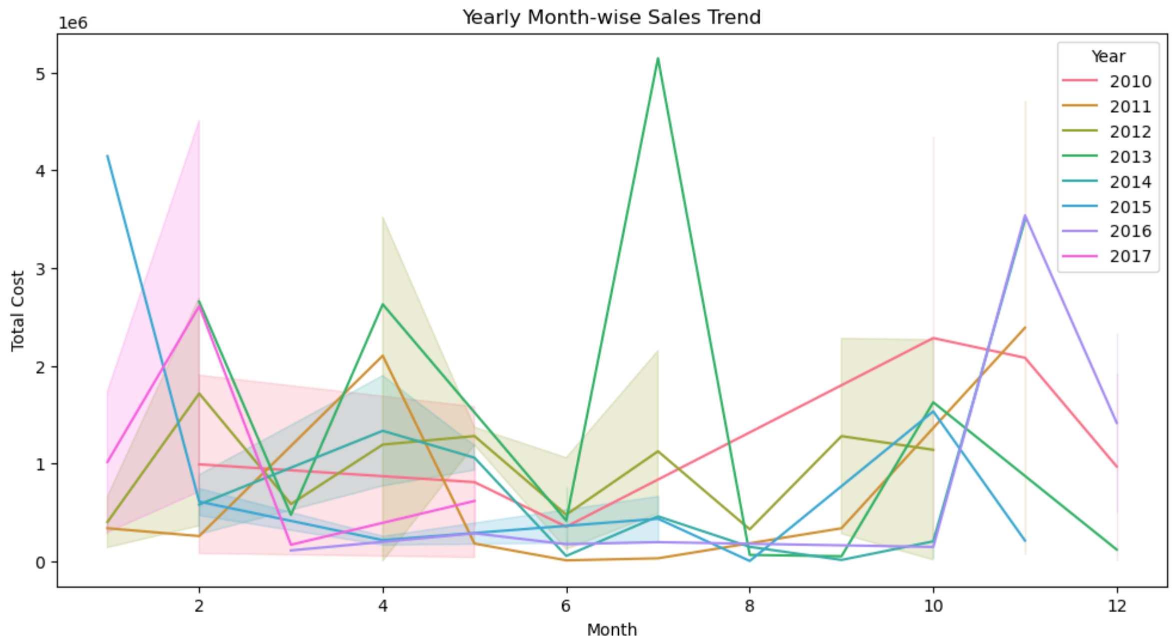
Yearly Month-wise Sales Trend of Item

In [10]:
```
yearly_monthly_sales = df.groupby(['year', 'month'])['Item Type'].sum()
yearly_monthly_sales
```

Out[10]:
```
year  month
2010  2                                  CosmeticsClothes
      5                                  Baby FoodFruits
      6                                          Clothes
      10                           ClothesOffice Supplies
      11                                       Cosmetics
      12                           HouseholdPersonal Care
2011  1                                  SnacksBeverages
      2                                        Beverages
      4                                        Household
      5                                        Beverages
      6                                       Vegetables
      7                                          Clothes
      9                                       Vegetables
      11        FruitsOffice SuppliesOffice SuppliesFruits
2012  1                          Office SuppliesHousehold
      2              Office SuppliesPersonal CareHousehold
      3                                       Vegetables
      4                      ClothesOffice SuppliesFruits
      5                                HouseholdBaby Food
      6                      CerealClothesOffice Supplies
      7                       VegetablesMeatPersonal Care
      8                                           Cereal
      9                                  CosmeticsClothes
      10                              VegetablesHousehold
2013  2                                  Office Supplies
      3                                           Cereal
      4                                  Office Supplies
      6                                  CerealBaby Food
      7                                CosmeticsCosmetics
      8                                           Fruits
      9                                           Fruits
      10                               CosmeticsCosmetics
      12                                    Personal Care
2014  2                           Personal CareBaby Food
      4                                  CerealCosmetics
      5                         Office SuppliesBaby Food
      6                                           Fruits
      7                                BeveragesBeverages
      8                                          Clothes
      9                                           Fruits
      10                          FruitsClothesBeverages
      11                                       Household
2015  1                                        Household
      2                               Baby FoodCosmetics
      4                                 BeveragesClothes
      7            Personal CareBaby FoodPersonal Care
      8                                           Fruits
      10                                 Office Supplies
      11                                         Clothes
2016  3                                           Cereal
      5                                    Personal Care
      6                                 SnacksVegetables
      7                                          Clothes
      10                                       Beverages
      11                              CosmeticsCosmetics
      12                         CosmeticsOffice Supplies
```

```
2017  1                                ClothesMeat
      2                            HouseholdSnacks
      3                              Personal Care
      5                 CosmeticsPersonal CareCereal
Name: Item Type, dtype: object
```

In [49]:
```
# Assuming 'yearly_monthly_sales' DataFrame is already calculated
# If not, use the provided 'groupby' operation to create it

# Define a custom color palette for each year
custom_palette = sns.color_palette("husl", n_colors=len(yearly_monthly_sales[

# Plotting the yearly month-wise sales trend
plt.figure(figsize=(12, 6))
sns.lineplot(x='month', y='Total Cost', hue='year', data=yearly_monthly_sales,
plt.title('Yearly Month-wise Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Cost')
plt.legend(title='Year')
plt.show()
```



Yearly Month-wise Sales Trend:

The yearly month-wise sales trend analysis allows for a detailed examination of sales patterns within each year. This can uncover specific months or seasons that contribute significantly to the overall sales performance.

In [11]:
```
# Step 5: Key Metrics and Factors
# Item-wise Total Cost In Year
product_total_cost_by_year = df.groupby(['year', 'Item Type'])['Total Cost'].s
product_total_cost_by_year
```

Out[11]:
```
year  Item Type
2010  Baby Food        1582243.50
      Clothes           655513.60
      Cosmetics        3987869.52
      Fruits             40288.24
      Household        1924728.20
                          ...
2017  Cosmetics         477943.95
      Household        4509793.96
      Meat             1738477.23
      Personal Care     534058.08
      Snacks            713942.88
Name: Total Cost, Length: 62, dtype: float64
```

In [12]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2010
item_total_cost_2010 = df[df['year'] == 2010].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2010
total_cost_2010 = df[df['year'] == 2010]['Total Cost'].sum()

# 3. Average Total Cost for 2010
average_cost_2010 = df[df['year'] == 2010]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2010:")
print(item_total_cost_2010)

print("\nYearly Total Cost for 2010:", total_cost_2010)
print("Average Total Cost for 2010:", average_cost_2010)
```

```
Item-wise Total Cost in Year 2010:
Item Type
Baby Food          1582243.50
Clothes             655513.60
Cosmetics          3987869.52
Fruits               40288.24
Household          1924728.20
Office Supplies    4350343.52
Personal Care        15470.91
Name: Total Cost, dtype: float64

Yearly Total Cost for 2010: 12556457.49
Average Total Cost for 2010: 1255645.7490000003
```

Item-wise Total Cost in Year 2010:

The breakdown of total cost by item type in the year 2010 and so on as well helps identify which products contribute the most to the overall costs. This insight can be used for strategic decision-making and inventory management.

In [13]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2011
item_total_cost_2011 = df[df['year'] == 2011].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2011
total_cost_2011 = df[df['year'] == 2011]['Total Cost'].sum()

# 3. Average Total Cost for 2011
average_cost_2011 = df[df['year'] == 2011]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2011:")
print(item_total_cost_2011)

print("\nYearly Total Cost for 2011:", total_cost_2011)
print("Average Total Cost for 2011:", average_cost_2011)
```

```
Item-wise Total Cost in Year 2011:
Item Type
Beverages           722459.54
Clothes              31825.92
Fruits               69552.92
Household          2104134.98
Office Supplies    4711516.00
Snacks              398042.40
Vegetables          350626.08
Name: Total Cost, dtype: float64

Yearly Total Cost for 2011: 8388157.84
Average Total Cost for 2011: 699013.1533333333
```

In [14]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2012
item_total_cost_2012 = df[df['year'] == 2012].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2012
total_cost_2012 = df[df['year'] == 2012]['Total Cost'].sum()

# 3. Average Total Cost for 2012
average_cost_2012 = df[df['year'] == 2012]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2012:")
print(item_total_cost_2012)

print("\nYearly Total Cost for 2012:", total_cost_2012)
print("Average Total Cost for 2012:", average_cost_2012)
```

```
Item-wise Total Cost in Year 2012:
Item Type
Baby Food          1373243.88
Cereal              576298.31
Clothes             467317.76
Cosmetics          2280701.13
Fruits                3612.24
Household          6297831.28
Meat               2154588.52
Office Supplies    7339990.72
Personal Care       854470.26
Vegetables         1337580.30
Name: Total Cost, dtype: float64

Yearly Total Cost for 2012: 22685634.4
Average Total Cost for 2012: 1031165.1999999997
```

In [15]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2013
item_total_cost_2013 = df[df['year'] == 2013].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2013
total_cost_2013 = df[df['year'] == 2013]['Total Cost'].sum()

# 3. Average Total Cost for 2013
average_cost_2013 = df[df['year'] == 2013]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2013:")
print(item_total_cost_2013)

print("\nYearly Total Cost for 2013:", total_cost_2013)
print("Average Total Cost for 2013:", average_cost_2013)
```

```
Item-wise Total Cost in Year 2013:
Item Type
Baby Food           757245.00
Cereal              555686.95
Cosmetics          6774954.24
Fruits              119321.56
Office Supplies    5287397.12
Personal Care       120423.75
Name: Total Cost, dtype: float64

Yearly Total Cost for 2013: 13615028.62
Average Total Cost for 2013: 1134585.7183333335
```

In [16]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2014
item_total_cost_2014 = df[df['year'] == 2014].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2010
total_cost_2014 = df[df['year'] == 2014]['Total Cost'].sum()

# 3. Average Total Cost for 2014
average_cost_2014 = df[df['year'] == 2014]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2014:")
print(item_total_cost_2014)

print("\nYearly Total Cost for 2014:", total_cost_2014)
print("Average Total Cost for 2014:", average_cost_2014)
```

```
Item-wise Total Cost in Year 2014:
Item Type
Baby Food          2073894.78
Beverages           759526.68
Cereal              772106.23
Clothes             430438.40
Cosmetics          1899925.95
Fruits              108554.04
Household          3494663.16
Office Supplies     933903.84
Personal Care       277739.67
Name: Total Cost, dtype: float64

Yearly Total Cost for 2014: 10750752.75
Average Total Cost for 2014: 716716.85
```

In [17]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2015
item_total_cost_2015 = df[df['year'] == 2015].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2015
total_cost_2015 = df[df['year'] == 2015]['Total Cost'].sum()

# 3. Average Total Cost for 2015
average_cost_2015 = df[df['year'] == 2015]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2015:")
print(item_total_cost_2015)

print("\nYearly Total Cost for 2015:", total_cost_2015)
print("Average Total Cost for 2015:", average_cost_2015)
```

```
Item-wise Total Cost in Year 2015:
Item Type
Baby Food           677056.74
Beverages           172619.70
Clothes             475668.48
Cosmetics           749700.51
Fruits                4657.16
Household          4145955.00
Office Supplies    1534983.04
Personal Care       670802.79
Name: Total Cost, dtype: float64

Yearly Total Cost for 2015: 8431443.42
Average Total Cost for 2015: 766494.8563636363
```

In [18]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2016
item_total_cost_2016 = df[df['year'] == 2016].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2016
total_cost_2016 = df[df['year'] == 2016]['Total Cost'].sum()

# 3. Average Total Cost for 2016
average_cost_2016 = df[df['year'] == 2016]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2016:")
print(item_total_cost_2016)

print("\nYearly Total Cost for 2016:", total_cost_2016)
print("Average Total Cost for 2016:", average_cost_2016)
```

```
Item-wise Total Cost in Year 2016:
Item Type
Beverages          148141.40
Cereal             112659.82
Clothes            197048.32
Cosmetics         5874365.64
Office Supplies    497662.08
Personal Care      287316.90
Snacks             216804.00
Vegetables         135031.05
Name: Total Cost, dtype: float64

Yearly Total Cost for 2016: 7469029.209999999
Average Total Cost for 2016: 746902.9210000001
```

In [19]:
```
# Assuming your dataset has a 'Total Cost' column
# You might adjust column names based on your actual dataset structure

# 1. Item-wise Total Cost in Year 2017
item_total_cost_2017 = df[df['year'] == 2017].groupby('Item Type')['Total Cost

# 2. Yearly Total Cost for 2016
total_cost_2017 = df[df['year'] == 2017]['Total Cost'].sum()

# 3. Average Total Cost for 2016
average_cost_2017 = df[df['year'] == 2017]['Total Cost'].mean()

# Display the results
print("Item-wise Total Cost in Year 2017:")
print(item_total_cost_2017)

print("\nYearly Total Cost for 2017:", total_cost_2017)
print("Average Total Cost for 2017:", average_cost_2017)
```

```
Item-wise Total Cost in Year 2017:
Item Type
Cereal           1013704.16
Clothes           296145.92
Cosmetics         477943.95
Household        4509793.96
Meat             1738477.23
Personal Care     534058.08
Snacks            713942.88
Name: Total Cost, dtype: float64

Yearly Total Cost for 2017: 9284066.18
Average Total Cost for 2017: 1160508.2725
```

Data Transformation

In [20]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['year'] = df['Order Date'].dt.year

# Iterate over each year
for year in range(2010, 2018):
    # Filter data for the specific year
    df_year = df[df['year'] == year]

    # Item-wise Total Cost for the year
    item_total_cost_by_year = df_year.groupby('Item Type')['Total Cost'].sum(

    # Display conclusions for the specific year
    print(f"\nConclusions for {year}:")

    # Example: Identify top-selling and low-selling items
    top_selling_item = item_total_cost_by_year.idxmax()
    low_selling_item = item_total_cost_by_year.idxmin()

    print(f"Top-selling item in {year}: {top_selling_item}")
    print(f"Lowest-selling item in {year}: {low_selling_item}")

    # Additional analyses or conclusions can be added based on your specific c
```

```
Conclusions for 2010:
Top-selling item in 2010: Office Supplies
Lowest-selling item in 2010: Personal Care

Conclusions for 2011:
Top-selling item in 2011: Office Supplies
Lowest-selling item in 2011: Clothes

Conclusions for 2012:
Top-selling item in 2012: Office Supplies
Lowest-selling item in 2012: Fruits

Conclusions for 2013:
Top-selling item in 2013: Cosmetics
Lowest-selling item in 2013: Fruits

Conclusions for 2014:
Top-selling item in 2014: Household
Lowest-selling item in 2014: Fruits

Conclusions for 2015:
Top-selling item in 2015: Household
Lowest-selling item in 2015: Fruits

Conclusions for 2016:
Top-selling item in 2016: Cosmetics
Lowest-selling item in 2016: Cereal

Conclusions for 2017:
Top-selling item in 2017: Household
Lowest-selling item in 2017: Clothes
```

Data Transformation

This shows the top selling item and lowest selling item of all the year.

In [21]:
```
# Year-wise Sales
yearly_sales = df.groupby('year')['Total Cost'].sum()
yearly_sales
```

Out[21]:
```
year
2010    12556457.49
2011     8388157.84
2012    22685634.40
2013    13615028.62
2014    10750752.75
2015     8431443.42
2016     7469029.21
2017     9284066.18
Name: Total Cost, dtype: float64
```

In [22]:
```
# Average Sales
average_sales = df.groupby('year')['Total Cost'].mean()
average_sales
```

Out[22]:
```
year
2010    1.255646e+06
2011    6.990132e+05
2012    1.031165e+06
2013    1.134586e+06
2014    7.167168e+05
2015    7.664949e+05
2016    7.469029e+05
2017    1.160508e+06
Name: Total Cost, dtype: float64
```

Year-wise Sales and Average Sales:

Calculating the total cost for each year and the average sales provides a high-level overview of the financial performance over time. It helps in understanding the overall revenue generation and stability.

In [23]:
```
# Step 6: Relationship Analysis
# Correlation matrix

df_numeric_no_na = df.select_dtypes(include=['float', 'int']).dropna()
correlation_matrix = df_numeric_no_na.corr()
correlation_matrix
```

Out[23]:

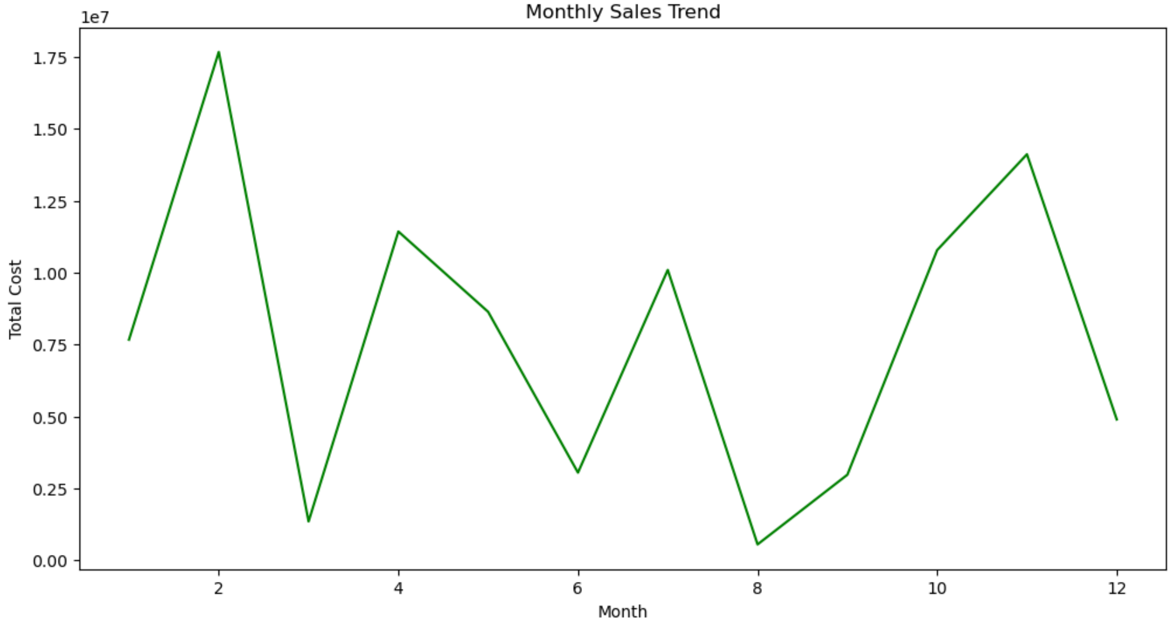
	Order ID	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit	year
Order ID	1.000000	-0.222907	-0.190941	-0.213201	-0.314688	-0.328944	-0.234638	0.081752
Units Sold	-0.222907	1.000000	-0.070486	-0.092232	0.447784	0.374746	0.564550	0.012455
Unit Price	-0.190941	-0.070486	1.000000	0.987270	0.752360	0.787905	0.557365	-0.061791
Unit Cost	-0.213201	-0.092232	0.987270	1.000000	0.715623	0.774895	0.467214	-0.071567
Total Revenue	-0.314688	0.447784	0.752360	0.715623	1.000000	0.983928	0.897327	-0.037128
Total Cost	-0.328944	0.374746	0.787905	0.774895	0.983928	1.000000	0.804091	-0.050899
Total Profit	-0.234638	0.564550	0.557365	0.467214	0.897327	0.804091	1.000000	0.002196
year	0.081752	0.012455	-0.061791	-0.071567	-0.037128	-0.050899	0.002196	1.000000
month	-0.111219	-0.007995	-0.031917	-0.042016	0.003835	-0.015617	0.051366	-0.106715

Correlation Matrix:

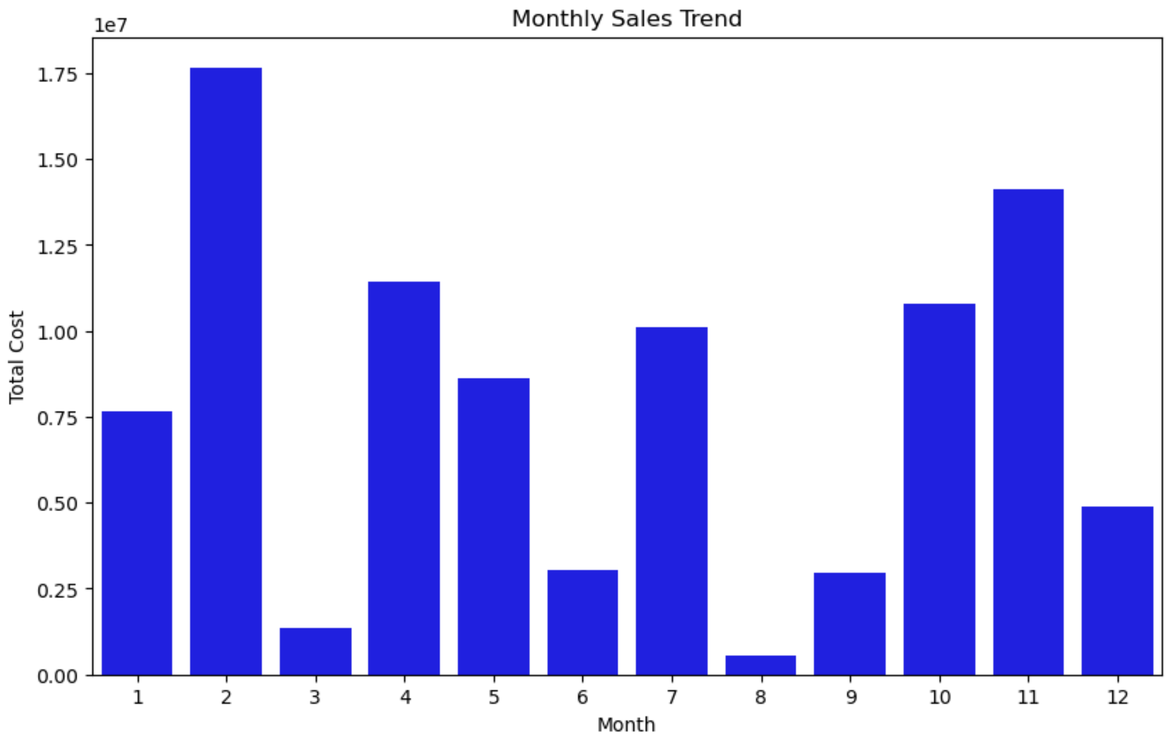
The correlation matrix provides information on the relationships between different variables. For example, it could highlight whether there are correlations between sales and other factors such as total cost or order date.

Graphical Representation

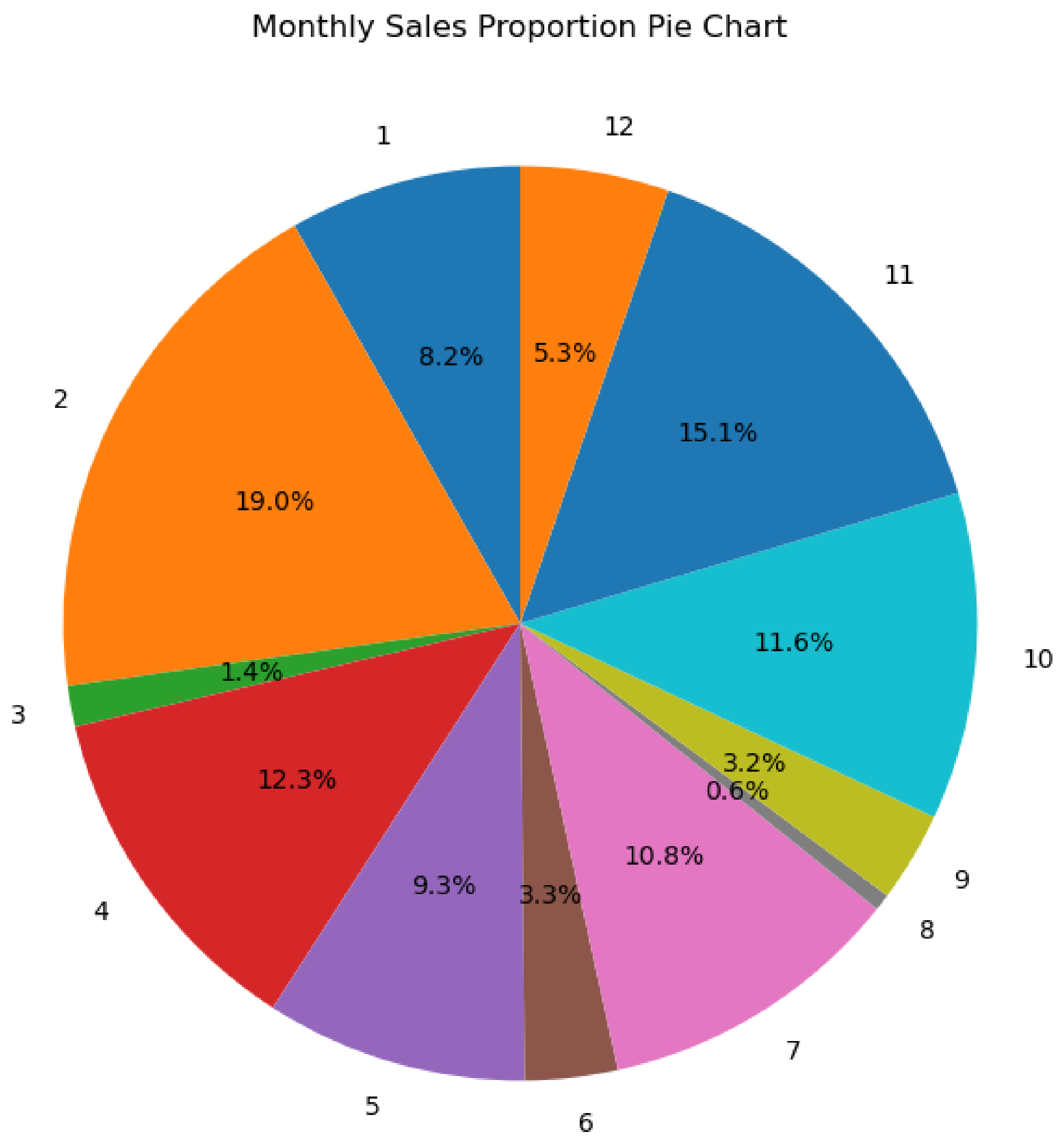
In [24]:
```
monthly_sales = df.groupby('month')['Total Cost'].sum()

plt.figure(figsize=(12, 6))
sns.lineplot(x=monthly_sales.index, y=monthly_sales.values, color='green')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Cost')
plt.show()
```



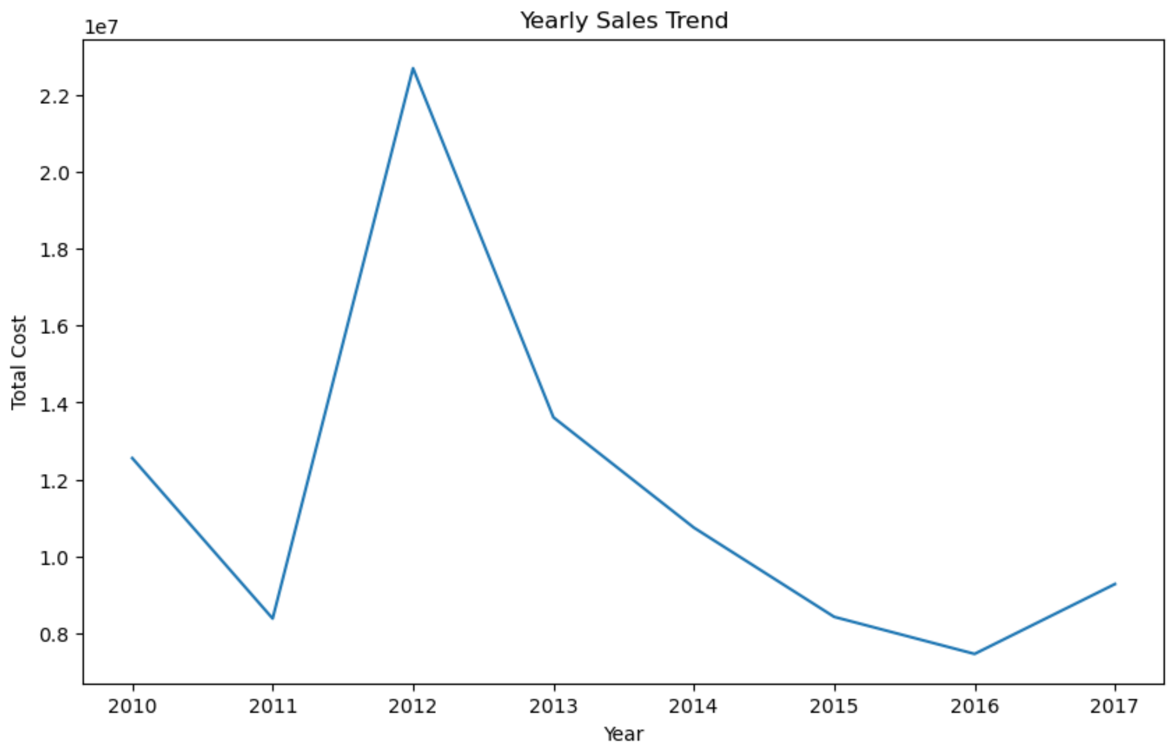
In [25]:
```
monthly_sales = df.groupby('month')['Total Cost'].sum()

plt.figure(figsize=(10, 6))
sns.barplot(x=monthly_sales.index, y=monthly_sales.values, color = 'blue')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Cost')
plt.show()
```



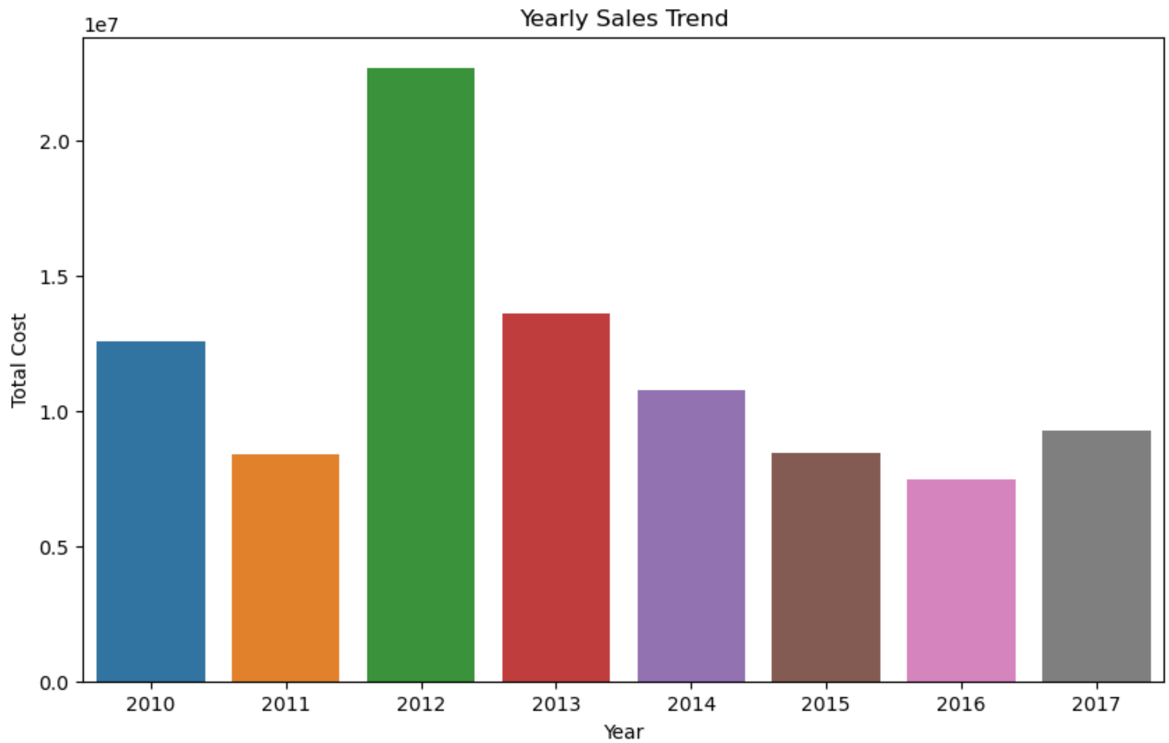
In [26]:
```
# Plot a pie chart
plt.figure(figsize=(8, 8))
plt.pie(monthly_sales, labels=monthly_sales.index, autopct='%1.1f%%', startang
plt.title('Monthly Sales Proportion Pie Chart')
plt.show()
```



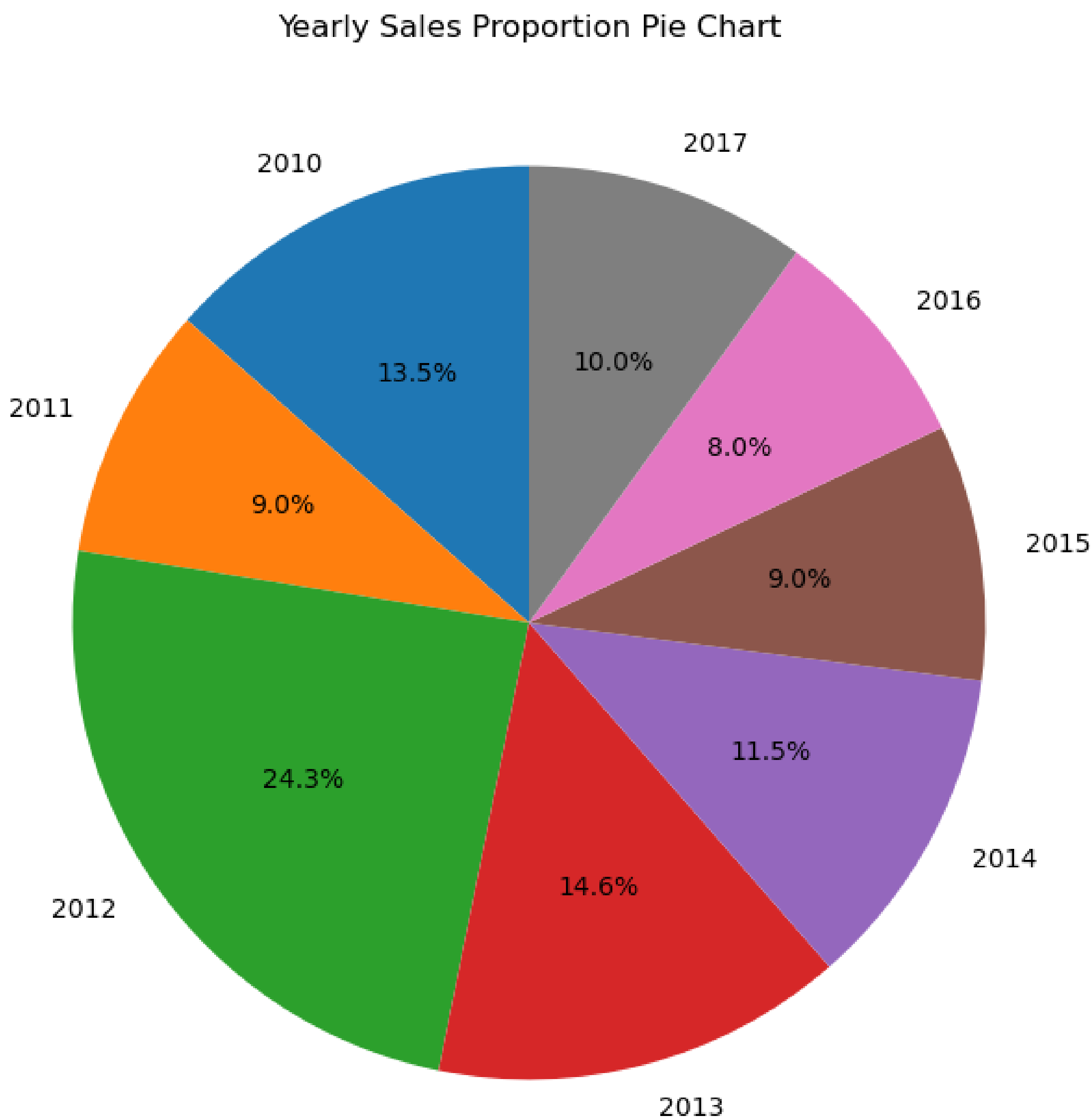
In [27]:
```
yearly_sales = df.groupby('year')['Total Cost'].sum()

plt.figure(figsize=(10, 6))
sns.lineplot(x=yearly_sales.index, y=yearly_sales.values)
plt.title('Yearly Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Cost')
plt.show()
```



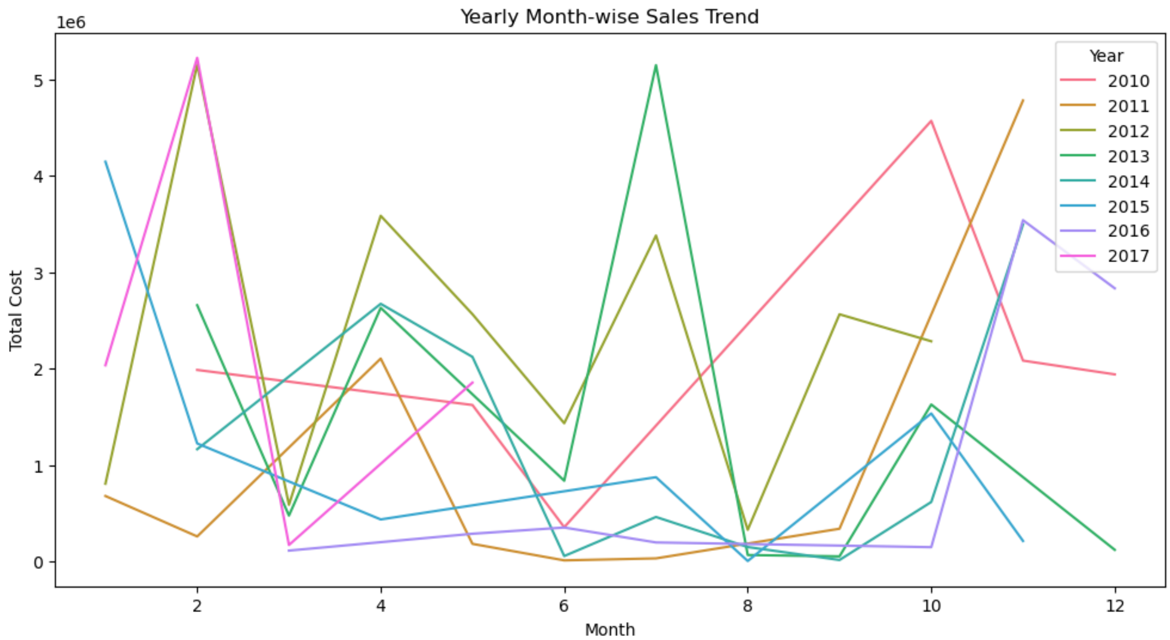
In [28]:
```
yearly_sales = df.groupby('year')['Total Cost'].sum()

plt.figure(figsize=(10, 6))
sns.barplot(x=yearly_sales.index, y=yearly_sales.values)
plt.title('Yearly Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Cost')
plt.show()
```



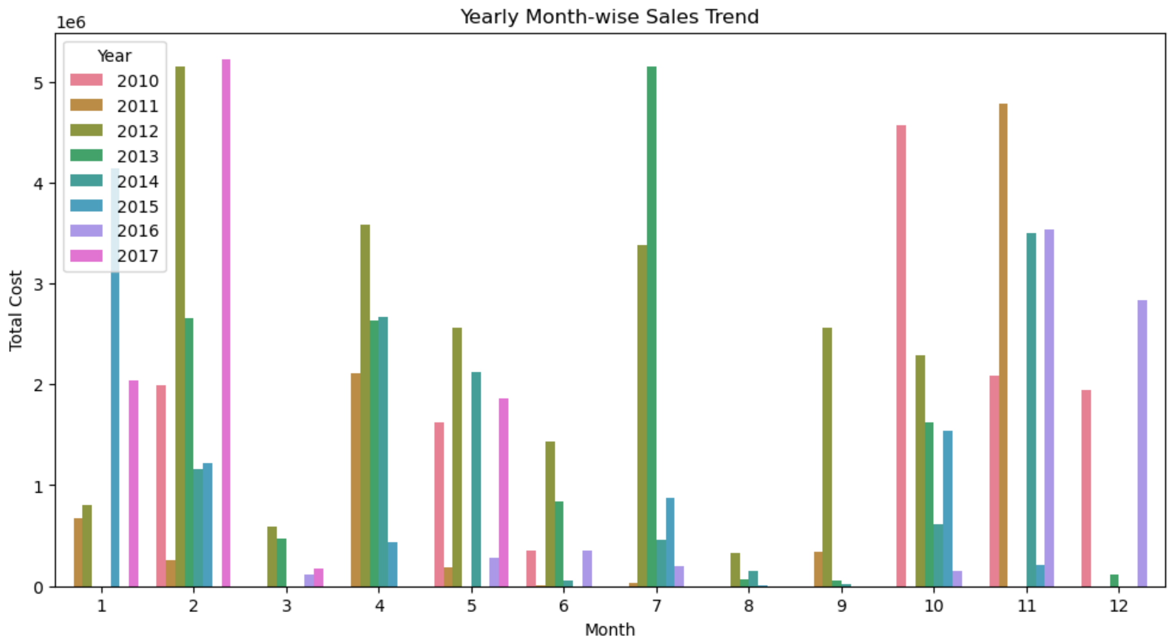
In [29]:
```
# Plot a pie chart
plt.figure(figsize=(8, 8))
plt.pie(yearly_sales, labels=yearly_sales.index, autopct='%1.1f%%', startangle
plt.title('Yearly Sales Proportion Pie Chart')
plt.show()
```



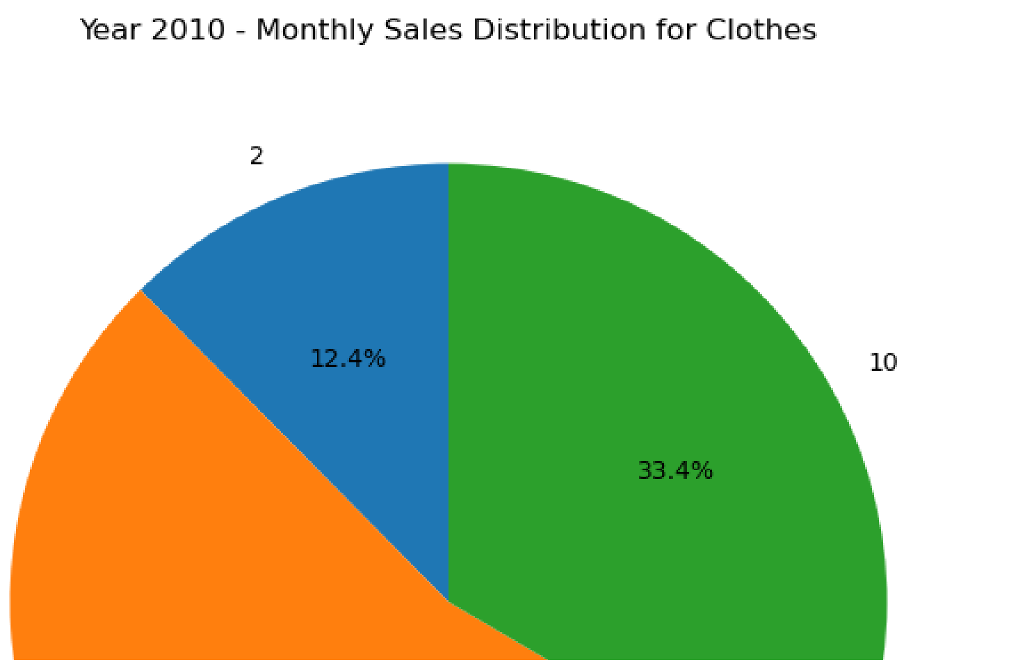
In [30]:
```
yearly_monthly_sales = df.groupby(['year', 'month'])['Total Cost'].sum().reset

# Define a custom color palette for each year
custom_palette = sns.color_palette("husl", n_colors=len(yearly_monthly_sales[

plt.figure(figsize=(12, 6))
sns.lineplot(x='month', y='Total Cost', hue='year', data=yearly_monthly_sales,
plt.title('Yearly Month-wise Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Cost')
plt.legend(title='Year')
plt.show()
```



In [31]:
```
yearly_monthly_sales = df.groupby(['year', 'month'])['Total Cost'].sum().reset

# Define a custom color palette for each year
custom_palette = sns.color_palette("husl", n_colors=len(yearly_monthly_sales[

plt.figure(figsize=(12, 6))
sns.barplot(x='month', y='Total Cost', hue='year', data=yearly_monthly_sales,p
plt.title('Yearly Month-wise Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Cost')
plt.legend(title='Year')
plt.show()
```



In [37]:
```
# Iterate over each year
for year in yearly_monthly_sales['year'].unique():
    # Iterate over each item type within the specific year
    for item_type in yearly_monthly_sales[(yearly_monthly_sales['year'] == yea
        # Filter data for the specific year and item type
        df_selected = yearly_monthly_sales[(yearly_monthly_sales['year'] == ye

        # Plot a pie chart for each year and item type
        plt.figure(figsize=(8, 8))
        plt.pie(df_selected['Total Cost'], labels=df_selected['month'], autopc
        plt.title(f'Year {year} - Monthly Sales Distribution for {item_type}')
        plt.show()
```



Overall Insights:

1. Sales Management Importance:

The analysis supports the notion that sales management is crucial for meeting competition, improving distribution methods, reducing costs, and increasing profits. By understanding sales trends and key metrics, businesses can make informed decisions to enhance their sales strategies.

2. Strategic Decision-Making:

The insights gained from the analysis can guide strategic decision-making. For instance, identifying top-performing products, understanding seasonal variations, and recognizing cost patterns can contribute to effective business planning.

Strategy

Here are some strategies to optimize methods of distribution, reduce costs, and increase profits:

1. Product Focus:

Identify the top-selling products and focus on optimizing their distribution channels. Ensure these products are well-stocked and readily available to meet customer demand. Consider bundling or promoting complementary products to increase sales and maximize revenue.

2. Seasonal Analysis:

Utilize the yearly month-wise sales trend analysis to understand seasonal variations in demand. Adjust inventory levels and distribution strategies accordingly to avoid overstocking or stockouts. Implement targeted marketing campaigns during peak seasons to boost sales and capitalize on high-demand periods.

3. Supply Chain Optimization:

Work on optimizing the supply chain to reduce distribution costs. This could involve negotiating better terms with suppliers, exploring more cost-effective shipping options, or implementing efficient inventory management systems. Utilize technology and data analytics to streamline the supply chain, reducing lead times and improving overall efficiency.

4. Cost Management:

Analyze the item-wise total cost, especially in the key year like 2010, to identify cost-intensive products. Explore opportunities for cost reduction through negotiations with suppliers, bulk purchasing, or finding alternative suppliers. Regularly review and optimize operating expenses to ensure cost-effectiveness in all aspects of the business.

5. Strategic Pricing:

Use the year-wise sales and average sales metrics to adjust pricing strategies. Consider dynamic pricing models that respond to market demand and competitor pricing. Implement promotions or discounts strategically to drive sales during slower periods or to clear excess inventory.

6. Customer Segmentation:

Leverage the correlation matrix insights to understand customer behavior and preferences. Tailor distribution strategies based on customer segments, ensuring personalized and targeted approaches. Implement loyalty programs or incentives to encourage repeat business and enhance customer lifetime value.

7. Technology Integration:

Embrace technology solutions to enhance distribution efficiency. This could involve adopting advanced inventory management systems, utilizing data analytics for demand forecasting, and implementing e-commerce platforms for seamless online sales. Explore automation options to reduce manual processes, improve accuracy, and decrease labor costs.

8. Continuous Improvement:

Regularly monitor and reassess the effectiveness of distribution methods. Stay agile and be willing to adapt strategies based on changing market conditions, customer preferences, and industry trends. Encourage a culture of continuous improvement within the organization, seeking feedback from customers and employees to identify areas for optimization. By combining these strategies and tailoring them to the specific insights derived from the analysis, businesses can create a robust plan to optimize distribution methods, reduce costs, and increase profits. Regular monitoring and adjustment based on performance metrics will be key to long-term success.

Conclusion

In conclusion, the analysis of sales trends, key metrics, and factors provides valuable insights that can be leveraged to formulate strategic decisions aimed at optimizing distribution methods, reducing costs, and increasing profits in a commercial and business enterprise. Here is a comprehensive summary of the conclusions and strategies:

Sales Trend Analysis:

Month-wise Sales Trend: Reveals seasonal patterns, aiding in inventory management and marketing strategies. Year-wise Sales Trend: Provides an overview of overall business performance in terms of total cost. Yearly Month-wise Sales Trend: Allows a detailed examination of sales patterns within each year.

Key Metrics and Factors:

Item-wise Total Cost in Year 2010: Identifies high-cost products, informing strategic decisions on inventory and cost management. Year-wise Sales and Average Sales: Offers a high-level overview of financial performance, aiding in understanding revenue generation and stability.

Relationship Analysis:

Correlation Matrix: Highlights relationships between variables, such as sales and other factors, guiding strategic decision-making.

Overall Insights:

Sales Management Importance: Affirms the significance of sales management in addressing competition, improving distribution, reducing costs, and increasing profits.

Strategic Decision-Making: Provides insights for strategic decisions, including product focus, seasonal adjustments, supply chain optimization, cost management, strategic pricing, customer segmentation, technology integration, and continuous improvement.

Strategies for Optimization:

Product Focus: Concentrate on top-selling products and consider bundling or promotions.

Seasonal Analysis: Adjust inventory and implement targeted marketing during peak seasons.

Supply Chain Optimization: Negotiate with suppliers, explore cost-effective shipping, and utilize technology for efficiency. Cost Management: Identify cost-intensive products and regularly review operating expenses.

Strategic Pricing: Adjust pricing strategies based on sales metrics and implement promotions strategically.

Customer Segmentation: Tailor distribution strategies based on customer behavior and preferences.

Technology Integration: Embrace technology solutions for efficient distribution and explore automation options.

Continuous Improvement: Monitor and adapt strategies based on changing market conditions, customer preferences, and industry trends.

In essence, businesses can achieve success by implementing a holistic approach that combines data-driven insights with strategic decision-making. Regular monitoring, adaptation, and a focus on continuous improvement will be crucial for long-term profitability and competitiveness.