**Analyzing Amazon Sales Date using Python**

**Exploratory Data Analysis (EDA) –**

*# import libraries*

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

In [2]:

*# Load the dataset*

df **=** pd.read\_csv(r"C:\Users\selff\OneDrive\Desktop\Unified Mentor Projects\Amazon Sales data\Amazon Sales data.csv")

df **=** pd.read\_csv(r"C:\Users\selff\OneDrive\Desktop\Unified Mentor Projects\Amazon Sales data\Amazon Sales data.csv")

In [3]:

*# Display the DataFrame*

df

Out[3]:

|  | **Region** | **Country** | **Item Type** | **Sales Channel** | **Order Priority** | **Order Date** | **Order ID** | **Ship Date** | **Units Sold** | **Unit Price** | **Unit Cost** | **Total Revenue** | **Total Cost** | **Total Profit** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Australia and Oceania | Tuvalu | Baby Food | Offline | H | 5/28/2010 | 669165933 | 6/27/2010 | 9925 | 255.28 | 159.42 | 2533654.00 | 1582243.50 | 951410.50 |
| **1** | Central America and the Caribbean | Grenada | Cereal | Online | C | 8/22/2012 | 963881480 | 9/15/2012 | 2804 | 205.70 | 117.11 | 576782.80 | 328376.44 | 248406.36 |
| **2** | Europe | Russia | Office Supplies | Offline | L | 5/2/2014 | 341417157 | 5/8/2014 | 1779 | 651.21 | 524.96 | 1158502.59 | 933903.84 | 224598.75 |
| **3** | Sub-Saharan Africa | Sao Tome and Principe | Fruits | Online | C | 6/20/2014 | 514321792 | 7/5/2014 | 8102 | 9.33 | 6.92 | 75591.66 | 56065.84 | 19525.82 |
| **4** | Sub-Saharan Africa | Rwanda | Office Supplies | Offline | L | 2/1/2013 | 115456712 | 2/6/2013 | 5062 | 651.21 | 524.96 | 3296425.02 | 2657347.52 | 639077.50 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **95** | Sub-Saharan Africa | Mali | Clothes | Online | M | 7/26/2011 | 512878119 | 9/3/2011 | 888 | 109.28 | 35.84 | 97040.64 | 31825.92 | 65214.72 |
| **96** | Asia | Malaysia | Fruits | Offline | L | 11/11/2011 | 810711038 | 12/28/2011 | 6267 | 9.33 | 6.92 | 58471.11 | 43367.64 | 15103.47 |
| **97** | Sub-Saharan Africa | Sierra Leone | Vegetables | Offline | C | 6/1/2016 | 728815257 | 6/29/2016 | 1485 | 154.06 | 90.93 | 228779.10 | 135031.05 | 93748.05 |
| **98** | North America | Mexico | Personal Care | Offline | M | 7/30/2015 | 559427106 | 8/8/2015 | 5767 | 81.73 | 56.67 | 471336.91 | 326815.89 | 144521.02 |
| **99** | Sub-Saharan Africa | Mozambique | Household | Offline | L | 2/10/2012 | 665095412 | 2/15/2012 | 5367 | 668.27 | 502.54 | 3586605.09 | 2697132.18 | 889472.91 |

100 rows × 14 columns

In [4]:

*# Display the columns of the DataFrame*

df.columns

Out[4]:

Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Priority',

'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit Price',

'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit'],

dtype='object')

In [6]:

*# Display the rows and columns of the DataFrame*

df.shape

Out[6]:

(100, 14)

In [8]:

*# Display total number of values in the DataFrame*

df.size

Out[8]:

1400

In [11]:

*# Display the columns information*

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 100 entries, 0 to 99

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Region 100 non-null object

1 Country 100 non-null object

2 Item Type 100 non-null object

3 Sales Channel 100 non-null object

4 Order Priority 100 non-null object

5 Order Date 100 non-null object

6 Order ID 100 non-null int64

7 Ship Date 100 non-null object

8 Units Sold 100 non-null int64

9 Unit Price 100 non-null float64

10 Unit Cost 100 non-null float64

11 Total Revenue 100 non-null float64

12 Total Cost 100 non-null float64

13 Total Profit 100 non-null float64

dtypes: float64(5), int64(2), object(7)

memory usage: 11.1+ KB

In [12]:

*# Display the Description of the DataFrame*

df.describe()

Out[12]:

|  | **Order ID** | **Units Sold** | **Unit Price** | **Unit Cost** | **Total Revenue** | **Total Cost** | **Total Profit** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 1.000000e+02 | 100.000000 | 100.000000 | 100.000000 | 1.000000e+02 | 1.000000e+02 | 1.000000e+02 |
| **mean** | 5.550204e+08 | 5128.710000 | 276.761300 | 191.048000 | 1.373488e+06 | 9.318057e+05 | 4.416820e+05 |
| **std** | 2.606153e+08 | 2794.484562 | 235.592241 | 188.208181 | 1.460029e+06 | 1.083938e+06 | 4.385379e+05 |
| **min** | 1.146066e+08 | 124.000000 | 9.330000 | 6.920000 | 4.870260e+03 | 3.612240e+03 | 1.258020e+03 |
| **25%** | 3.389225e+08 | 2836.250000 | 81.730000 | 35.840000 | 2.687212e+05 | 1.688680e+05 | 1.214436e+05 |
| **50%** | 5.577086e+08 | 5382.500000 | 179.880000 | 107.275000 | 7.523144e+05 | 3.635664e+05 | 2.907680e+05 |
| **75%** | 7.907551e+08 | 7369.000000 | 437.200000 | 263.330000 | 2.212045e+06 | 1.613870e+06 | 6.358288e+05 |
| **max** | 9.940222e+08 | 9925.000000 | 668.270000 | 524.960000 | 5.997055e+06 | 4.509794e+06 | 1.719922e+06 |

In [14]:

*#Display the null values for each column*

df.isnull().sum()

Out[14]:

Region 0

Country 0

Item Type 0

Sales Channel 0

Order Priority 0

Order Date 0

Order ID 0

Ship Date 0

Units Sold 0

Unit Price 0

Unit Cost 0

Total Revenue 0

Total Cost 0

Total Profit 0

dtype: int64

In [21]:

In [17]:

*# DataFrame is grouped by Item Type*

Item\_types **=** df.groupby('Item Type')

In [20]:

print(Item\_types.groups)

{'Baby Food': [0, 5, 20, 21, 61, 78, 87], 'Beverages': [17, 28, 72, 76, 86, 89, 91, 94], 'Cereal': [1, 9, 42, 45, 52, 60, 73], 'Clothes': [11, 12, 15, 44, 48, 50, 55, 56, 63, 67, 69, 81, 95], 'Cosmetics': [16, 25, 30, 41, 46, 57, 59, 64, 65, 74, 79, 82, 93], 'Fruits': [3, 23, 26, 27, 36, 51, 54, 71, 88, 96], 'Household': [6, 13, 18, 29, 33, 58, 75, 80, 99], 'Meat': [19, 37], 'Office Supplies': [2, 4, 32, 38, 39, 49, 53, 62, 68, 70, 90, 92], 'Personal Care': [8, 14, 24, 31, 35, 40, 47, 66, 85, 98], 'Snacks': [22, 34, 83], 'Vegetables': [7, 10, 43, 77, 84, 97]}

In [27]:

*# Convert Order Date column to datetime format*

df['Order Date'] **=** pd.to\_datetime(df['Order Date'])

In [28]:

In [29]:

*# Convert the Ship Date column to datetime format*

df['Ship Date'] **=** pd.to\_datetime(df['Ship Date'])

In [30]:

*# Display the information of DataFrame*

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 100 entries, 0 to 99

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Region 100 non-null object

1 Country 100 non-null object

2 Item Type 100 non-null object

3 Sales Channel 100 non-null object

4 Order Priority 100 non-null object

5 Order Date 100 non-null datetime64[ns]

6 Order ID 100 non-null int64

7 Ship Date 100 non-null datetime64[ns]

8 Units Sold 100 non-null int64

9 Unit Price 100 non-null float64

10 Unit Cost 100 non-null float64

11 Total Revenue 100 non-null float64

12 Total Cost 100 non-null float64

13 Total Profit 100 non-null float64

dtypes: datetime64[ns](2), float64(5), int64(2), object(5)

memory usage: 11.1+ KB

In [32]:

*# Display total number of duplicate values*

print(df.duplicated().sum())

0

*# Gives total number of Units Sold*

Total\_units\_sold **=** df['Units Sold'].sum()

In [36]:

print(Total\_units\_sold)

512871

In [37]:

*# Extract month from 'Order Date'*

df['Month'] **=** df['Order Date'].dt.month

In [39]:

*# Extract year from 'Order Date'*

df['Year'] **=** df['Order Date'].dt.year

In [40]:

*# Extract Year-month from 'Order Date'*

df['Year\_Month'] **=** df['Order Date'].dt.to\_period('M')

In [43]:

*#Monthly Sales Trend*

monthly\_sales\_trend **=** df.groupby('Month')['Total Revenue'].sum()

In [44]:

print(monthly\_sales\_trend)

Month

1 10482467.12

2 24740517.77

3 2274823.87

4 16187186.33

5 13215739.99

6 5230325.77

7 15669518.50

8 1128164.91

9 5314762.56

10 15287576.61

11 20568222.76

12 7249462.12

Name: Total Revenue, dtype: float64

In [51]:

print(monthly\_sales\_trend.min())

print(monthly\_sales\_trend.max())

1128164.91

24740517.77

In [50]:

monthly\_sales\_trend.plot(kind**=**'bar', xlabel**=**'Month', ylabel**=**'Total Revenue', title**=**'Monthly Sales Trend')

Out[50]:

<AxesSubplot:title={'center':'Monthly Sales Trend'}, xlabel='Month', ylabel='Total Revenue'>

A blue bars on a white background

Description automatically generated

In [58]:

*# Yearly Sales Trend*

yearly\_sales\_trend **=** df.groupby('Year')['Total Revenue'].sum()

In [57]:

print(yearly\_sales\_trend.min())

print(yearly\_sales\_trend.max())

11129166.07

31898644.52

In [59]:

print(yearly\_sales\_trend)

Year

2010 19186024.92

2011 11129166.07

2012 31898644.52

2013 20330448.66

2014 16630214.43

2015 12427982.86

2016 12372867.22

2017 13373419.63

Name: Total Revenue, dtype: float64

In [64]:

yearly\_sales\_trend.plot(kind**=**'bar',xlabel**=**'Year', ylabel**=**'Total Revenue', title**=**'Yearly Sales Trend')

plt.show()

A blue bars on a white background

Description automatically generated

In [61]:

*# Yearly-Monthly Sales Trend*

yearly\_monthly\_sales\_trend **=** df.groupby('Year\_Month')['Total Revenue'].sum()

In [62]:

yearly\_monthly\_sales\_trend.plot(kind**=**'bar', xlabel**=**'Year-Month', ylabel**=**'Total Revenue', title**=**'Yearly-Monthly Sales Trend')

plt.show()

A blue lines on a white background

Description automatically generated

In [65]:

*# Find key metrics and factors*

key\_metrics **=** df[['Units Sold', 'Unit Price', 'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit']]

In [66]:

*# Show meaningful relationships between attributes*

correlation\_matrix **=** key\_metrics.corr()

print("Correlation Matrix:")

print(correlation\_matrix)

Correlation Matrix:

Units Sold Unit Price Unit Cost Total Revenue Total Cost \

Units Sold 1.000000 -0.070486 -0.092232 0.447784 0.374746

Unit Price -0.070486 1.000000 0.987270 0.752360 0.787905

Unit Cost -0.092232 0.987270 1.000000 0.715623 0.774895

Total Revenue 0.447784 0.752360 0.715623 1.000000 0.983928

Total Cost 0.374746 0.787905 0.774895 0.983928 1.000000

Total Profit 0.564550 0.557365 0.467214 0.897327 0.804091

Total Profit

Units Sold 0.564550

Unit Price 0.557365

Unit Cost 0.467214

Total Revenue 0.897327

Total Cost 0.804091

Total Profit 1.000000

​