Project - Market Risk Analysis

Part A

Problem Statement

An automobile parts manufacturing company has collected data on transactions for 3 years. They do not have any in-house data science team, thus they have hired you as their consultant. Your job is to use your data science skills to find the underlying buying patterns of the customers, provide the company with suitable insights about their customers, and recommend customized marketing strategies for different segments of customers.

Data Dictionary

ORDERNUMBER: This column represents the unique identification number assigned to each order.
QUANTITYORDERED: It indicates the number of items ordered in each order.
PRICEEACH: This column specifies the price of each item in the order.
ORDERLINENUMBER: It represents the line number of each item within an order.
SALES: This column denotes the total sales amount for each order, which is calculated by multiplying the quantity ordered by the price of each item.
ORDERDATE: It denotes the date on which the order was placed.
DAYS_SINCE_LASTORDER: This column represents the number of days that have passed since the last order for each customer. It can be used to analyze customer purchasing patterns.
STATUS: It indicates the status of the order, such as "Shipped," "In Process," "Cancelled," "Disputed," "On Hold," or "Resolved"
PRODUCTLINE: This column specifies the product line categories to which each item belongs.
MSRP: It stands for Manufacturer's Suggested Retail Price and represents the suggested selling price for each item.
PRODUCTCODE: This column represents the unique code assigned to each product.
CUSTOMERNAME: It denotes the name of the customer who placed the order.
PHONE: This column contains the contact phone number for the customer.
ADDRESSLINE1: It represents the first line of the customer's address.
CITY: This column specifies the city where the customer is located.
POSTALCODE: It denotes the postal code or ZIP code associated with the customer's address.
COUNTRY: This column indicates the country where the customer is located.
CONTACTLASTNAME: It represents the last name of the contact person associated with the customer.
CONTACTFIRSTNAME: This column denotes the first name of the contact person associated

with the customer.
DEALSIZE: It indicates the size of the deal or order, which are the categories "Small," "Medium," or "Large."

Importing required libraries

In [1]:
```
# Import libraries for data manipulation
import numpy as np
import pandas as pd

# Import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

Understanding the structure of data

In [2]:
```
df_sales = pd.read_excel('sales_data.xlsx')

df_sales.head() # Returns first 5 rows
```

Out[2]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDEF
0	10107	30	95.70	2	2871.00	2018-
1	10121	34	81.35	5	2765.90	2018-
2	10134	41	94.74	2	3884.34	2018-
3	10145	45	83.26	6	3746.70	2018-
4	10168	36	96.66	1	3479.76	2018-

Number of rows and columns in the dataset

In [3]:
```
# checking shape of the data

rows = str(df_sales.shape[0])
columns = str(df_sales.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

```
There are 2747 rows and 20 columns in the dataset.
```

Datatypes of the different columns in the dataset

```
In [4]: df_sales.info() # Concise summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2747 entries, 0 to 2746
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   ORDERNUMBER           2747 non-null   int64         
 1   QUANTITYORDERED       2747 non-null   int64         
 2   PRICEEACH             2747 non-null   float64       
 3   ORDERLINENUMBER       2747 non-null   int64         
 4   SALES                 2747 non-null   float64       
 5   ORDERDATE             2747 non-null   datetime64[ns]
 6   DAYS_SINCE_LASTORDER  2747 non-null   int64         
 7   STATUS                2747 non-null   object        
 8   PRODUCTLINE           2747 non-null   object        
 9   MSRP                  2747 non-null   int64         
 10  PRODUCTCODE           2747 non-null   object        
 11  CUSTOMERNAME          2747 non-null   object        
 12  PHONE                 2747 non-null   object        
 13  ADDRESSLINE1          2747 non-null   object        
 14  CITY                  2747 non-null   object        
 15  POSTALCODE            2747 non-null   object        
 16  COUNTRY               2747 non-null   object        
 17  CONTACTLASTNAME       2747 non-null   object        
 18  CONTACTFIRSTNAME      2747 non-null   object        
 19  DEALSIZE              2747 non-null   object        
dtypes: datetime64[ns](1), float64(2), int64(5), object(12)
memory usage: 429.3+ KB
```

There are 19 columns in the dataset. Out of which 1 have datetime data type, 2 have float data type, 5 have integer data type and 12 have object data type.

Finding missing values in the dataset

```
In [5]: df_sales.isna().sum()  # Count NaN values in all columns of dataset
```

Out[5]:
```
ORDERNUMBER            0
QUANTITYORDERED        0
PRICEEACH              0
ORDERLINENUMBER        0
SALES                  0
ORDERDATE              0
DAYS_SINCE_LASTORDER   0
STATUS                 0
PRODUCTLINE            0
MSRP                   0
PRODUCTCODE            0
CUSTOMERNAME           0
PHONE                  0
ADDRESSLINE1           0
CITY                   0
POSTALCODE             0
COUNTRY                0
CONTACTLASTNAME        0
CONTACTFIRSTNAME       0
DEALSIZE               0
dtype: int64
```

There are no missing values in the dataset.

Checking for Duplicates

In [6]:
```
df_sales.duplicated().sum() # Checking for duplicates
```

Out[6]:
```
0
```

There are no duplicate rows in the dataset.

Statistical summary of the data

In [7]:
```
# Summary statistics of the numerical data

df_sales[['QUANTITYORDERED','PRICEEACH','SALES','DAYS_SINCE_LASTORDER','MSRP']].des
```

Out[7]:

	count	mean	std	min	25%	50%
QUANTITYORDERED	2747.0	35.103021	9.762135	6.00	27.000	35.00	43
PRICEEACH	2747.0	101.098951	42.042548	26.88	68.745	95.55	127
SALES	2747.0	3553.047583	1838.953901	482.13	2204.350	3184.80	4503
DAYS_SINCE_LASTORDER	2747.0	1757.085912	819.280576	42.00	1077.000	1761.00	2436
MSRP	2747.0	100.691664	40.114802	33.00	68.000	99.00	124

Observations and Insights:

1. Minimum number of items ordered in each order is 6. Maximum number of items ordered in each order is 97. Average number of items ordered in each order is 35.
2. Minimum price of each item in the order is 26.88. Maximum price of each item in the order is 252.87. Average price of each item in the order is 101.10.
3. Minimum total sales amount for each order is 482.13. Maximum total sales amount for each order is 14082.80. Average total sales amount for each order is 3553.05.
4. Minimum number of days that have passed since the last order for each customer is 42. Maximum number of days that have passed since the last order for each customer is 3562. Average number of days that have passed since the last order for each customer is 1757.
5. Minimum Manufacturer's Suggested Retail Price is 33.00. Maximum Manufacturer's Suggested Retail Price is 214.00. Average Manufacturer's Suggested Retail Price is 100.69.
6. Total sales increase when number of items and price of each item increase.

Exploratory Data Analysis (EDA)

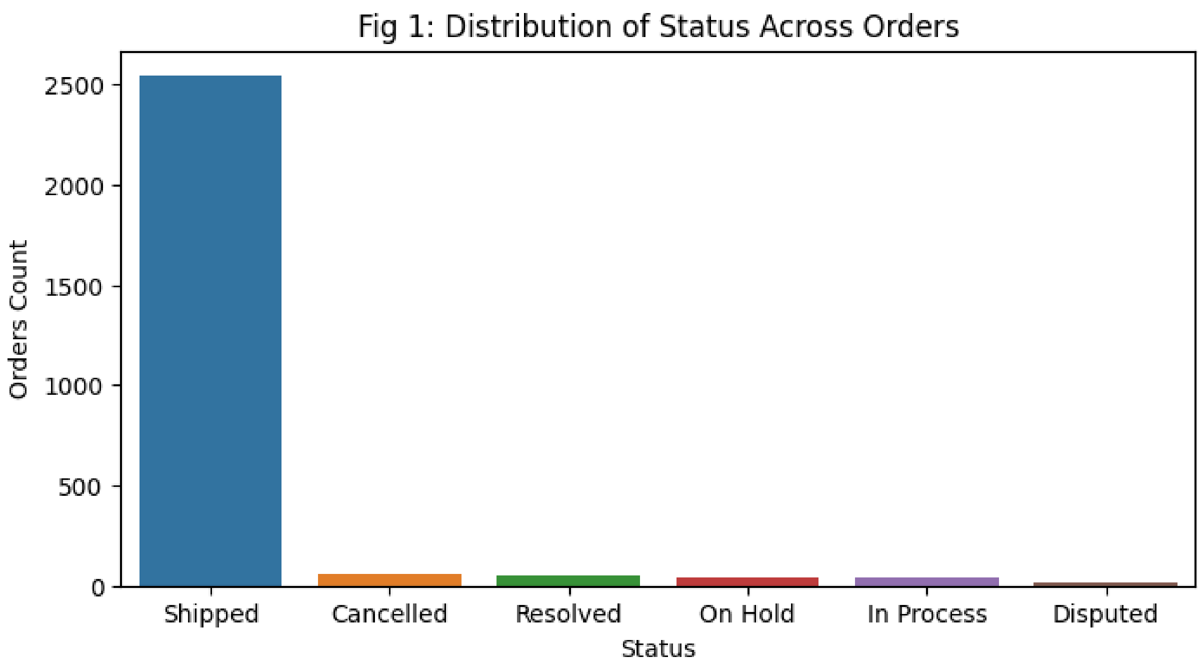
Univariate Analysis

STATUS

In [8]:
```
# Check unique STATUS

df_sales['STATUS'].value_counts() # Frequency of each distinct value in the STATUS
```

Out[8]:
```
STATUS
Shipped       2541
Cancelled       60
Resolved        47
On Hold         44
In Process      41
Disputed        14
Name: count, dtype: int64
```

In [9]:
```
# Count Plot - Distribution of Status across orders

plt.figure(figsize=(8,4))
sns.countplot(data=df_sales, x='STATUS', order = df_sales['STATUS'].value_counts().
plt.title('Fig 1: Distribution of Status Across Orders')
plt.xlabel('Status')
plt.ylabel('Orders Count')
plt.show()
```

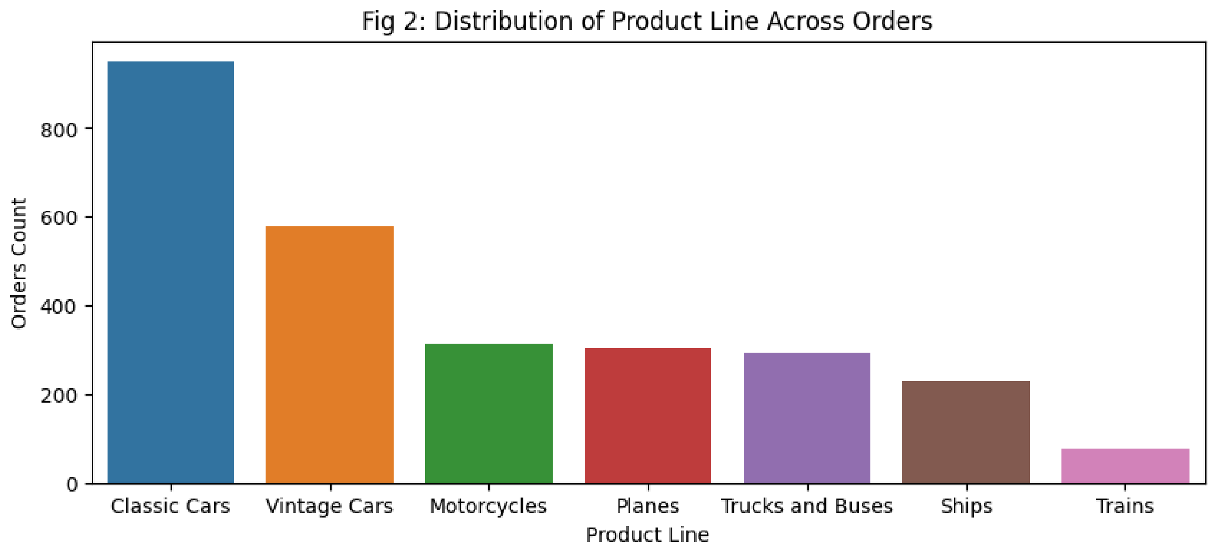


PRODUCTLINE

In [10]:
```
# check unique PRODUCTLINE

df_sales['PRODUCTLINE'].value_counts() # Frequency of each distinct value in the PR
```

Out[10]:
```
PRODUCTLINE
Classic Cars        949
Vintage Cars        579
Motorcycles         313
Planes              304
Trucks and Buses    295
Ships               230
Trains               77
Name: count, dtype: int64
```

In [11]:
```
# Count Plot - Distribution of Product Line across orders

plt.figure(figsize=(10,4))
sns.countplot(data=df_sales, x='PRODUCTLINE', order = df_sales['PRODUCTLINE'].value
plt.title('Fig 2: Distribution of Product Line Across Orders')
plt.xlabel('Product Line')
plt.ylabel('Orders Count')
plt.show()
```

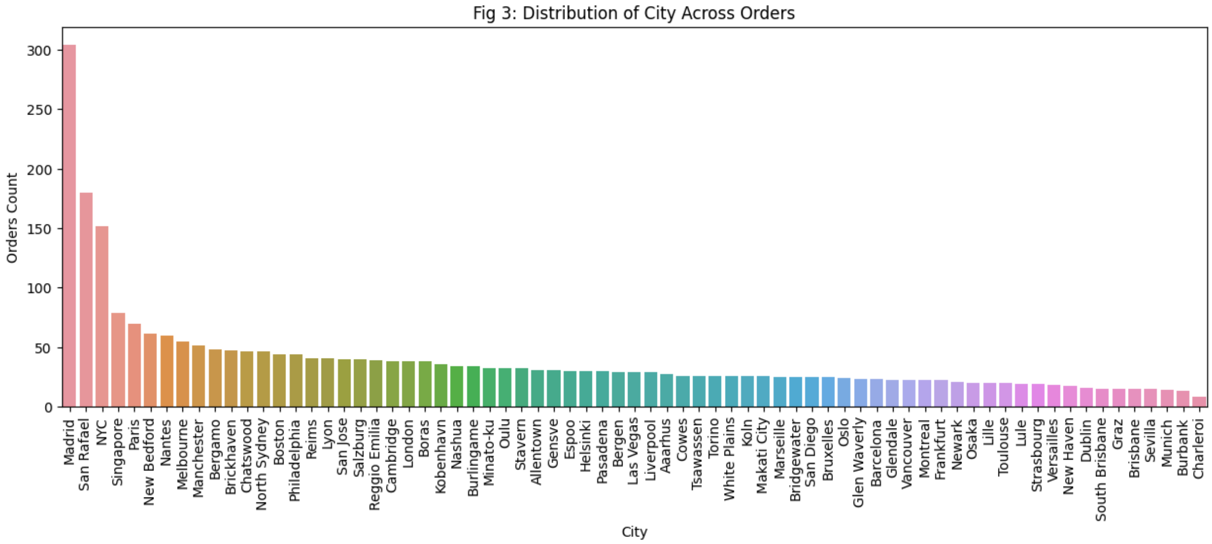


CITY

In [12]:
```
# check unique CITY

df_sales['CITY'].value_counts() # Frequency of each distinct value in the CITY colu
```

Out[12]:
```
CITY
Madrid         304
San Rafael     180
NYC            152
Singapore       79
Paris           70
              ...
Brisbane        15
Sevilla         15
Munich          14
Burbank         13
Charleroi        8
Name: count, Length: 71, dtype: int64
```

In [13]:
```
# Count Plot - Distribution of City across orders

plt.figure(figsize=(15,5))
sns.countplot(data=df_sales, x='CITY', order = df_sales['CITY'].value_counts().inde
plt.title('Fig 3: Distribution of City Across Orders')
plt.xticks(rotation=90)
plt.xlabel('City')
plt.ylabel('Orders Count')
plt.show()
```

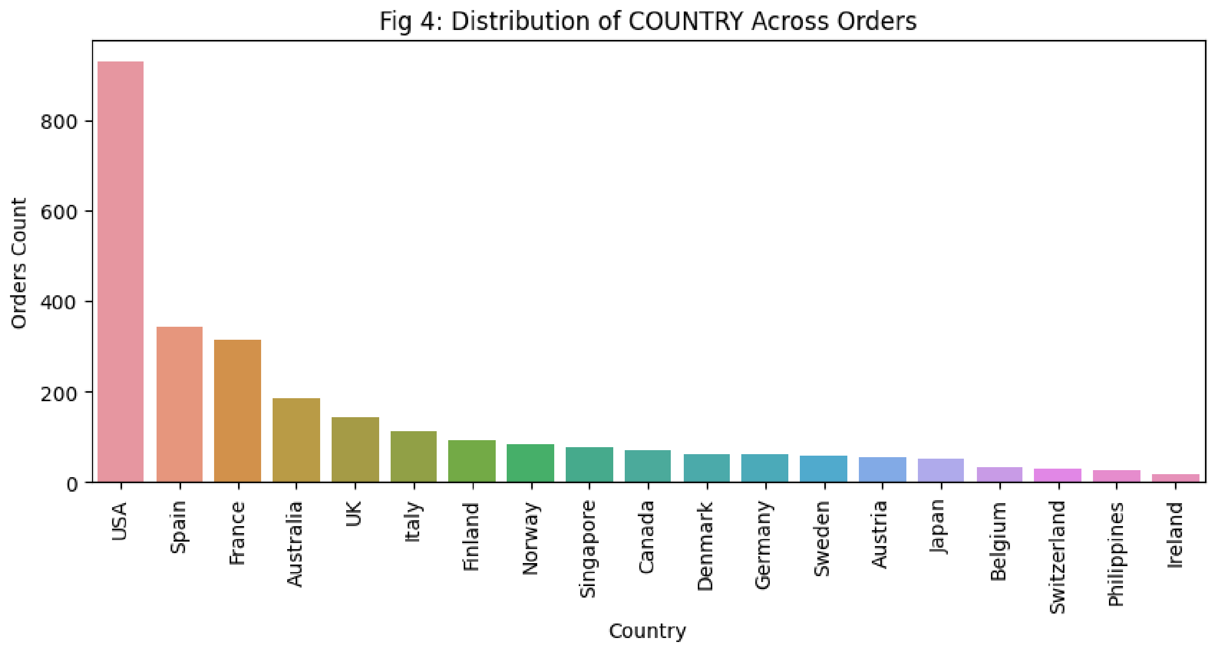


COUNTRY

In [14]:
```
# Check unique COUNTRY

df_sales['COUNTRY'].value_counts() # Frequency of each distinct value in the COUNTR
```

Out[14]:
```
COUNTRY
USA            928
Spain          342
France         314
Australia      185
UK             144
Italy          113
Finland         92
Norway          85
Singapore       79
Canada          70
Denmark         63
Germany         62
Sweden          57
Austria         55
Japan           52
Belgium         33
Switzerland     31
Philippines     26
Ireland         16
Name: count, dtype: int64
```

In [15]:
```
# Count Plot - Distribution of COUNTRY across orders

plt.figure(figsize=(10,4))
sns.countplot(data=df_sales, x='COUNTRY', order = df_sales['COUNTRY'].value_counts(
plt.title('Fig 4: Distribution of COUNTRY Across Orders')
plt.xticks(rotation=90)
plt.xlabel('Country')
plt.ylabel('Orders Count')
plt.show()
```

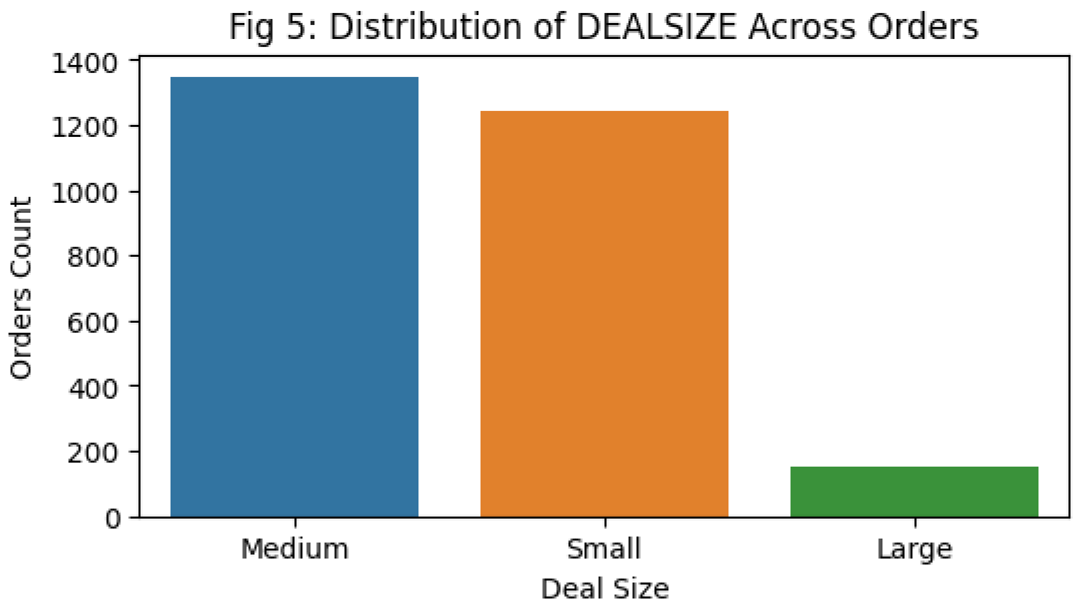


DEALSIZE

In [16]:
```
# Check unique DEALSIZE

df_sales['DEALSIZE'].value_counts() # Frequency of each distinct value in the DEALS
```

Out[16]:
```
DEALSIZE
Medium    1349
Small     1246
Large      152
Name: count, dtype: int64
```

In [17]:
```
# Count Plot - Distribution of DEALSIZE across orders

plt.figure(figsize=(6,3))
sns.countplot(data=df_sales, x='DEALSIZE', order = df_sales['DEALSIZE'].value_count
plt.title('Fig 5: Distribution of DEALSIZE Across Orders')
plt.xlabel('Deal Size')
plt.ylabel('Orders Count')
plt.show()
```

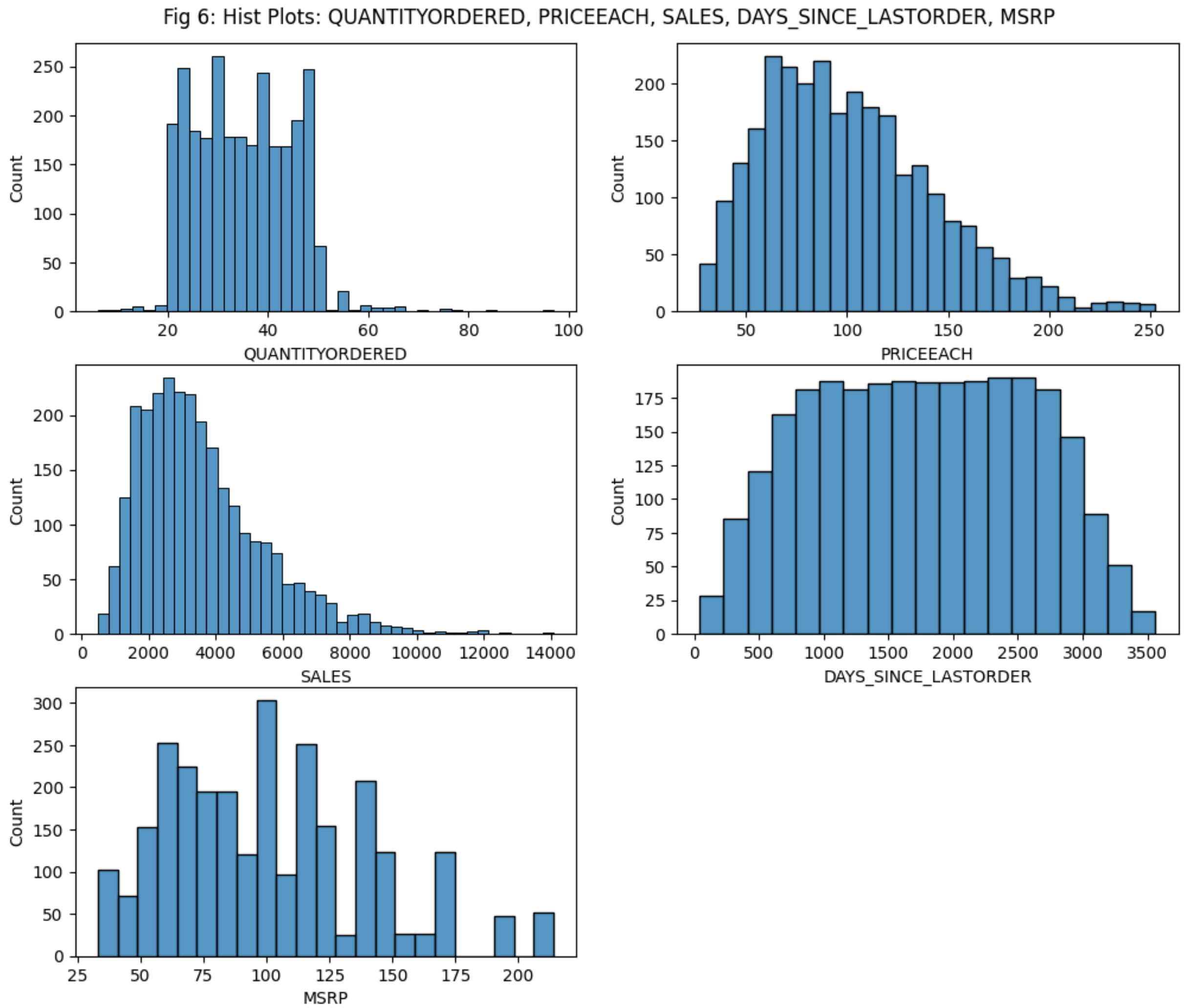


Observations and Insights:

1. Maximum orders have Shipped as order status while Disputed is the lowest.
2. Maximum orders have Classic Cars as product line while Trains is the lowest.
3. Madrid city has highest number of orders while Charleroi city has lowest number of orders.
4. USA country has highest number of orders while Ireland country has lowest number of orders.
5. Maximum orders have Medium as deal size while Large is the lowest.

In [18]:

```
# Hist Plots for QUANTITYORDERED, PRICEEACH, SALES, DAYS_SINCE_LASTORDER and MSRP

fig, axes = plt.subplots(3,2, figsize=(12, 10))

sns.histplot(ax=axes[0, 0], data=df_sales, x='QUANTITYORDERED')
sns.histplot(ax=axes[0, 1], data=df_sales, x='PRICEEACH')
sns.histplot(ax=axes[1, 0], data=df_sales, x='SALES')
sns.histplot(ax=axes[1, 1], data=df_sales, x='DAYS_SINCE_LASTORDER')
sns.histplot(ax=axes[2, 0], data=df_sales, x='MSRP')
axes[2,1].axis("off")

axes[0, 0].set(xlabel='QUANTITYORDERED')
axes[0, 1].set(xlabel='PRICEEACH')
axes[1, 0].set(xlabel='SALES')
axes[1, 1].set(xlabel='DAYS_SINCE_LASTORDER')
axes[2, 0].set(xlabel='MSRP')

plt.suptitle('Fig 6: Hist Plots: QUANTITYORDERED, PRICEEACH, SALES, DAYS_SINCE_LAST

plt.show()
```



Observations and Insights:

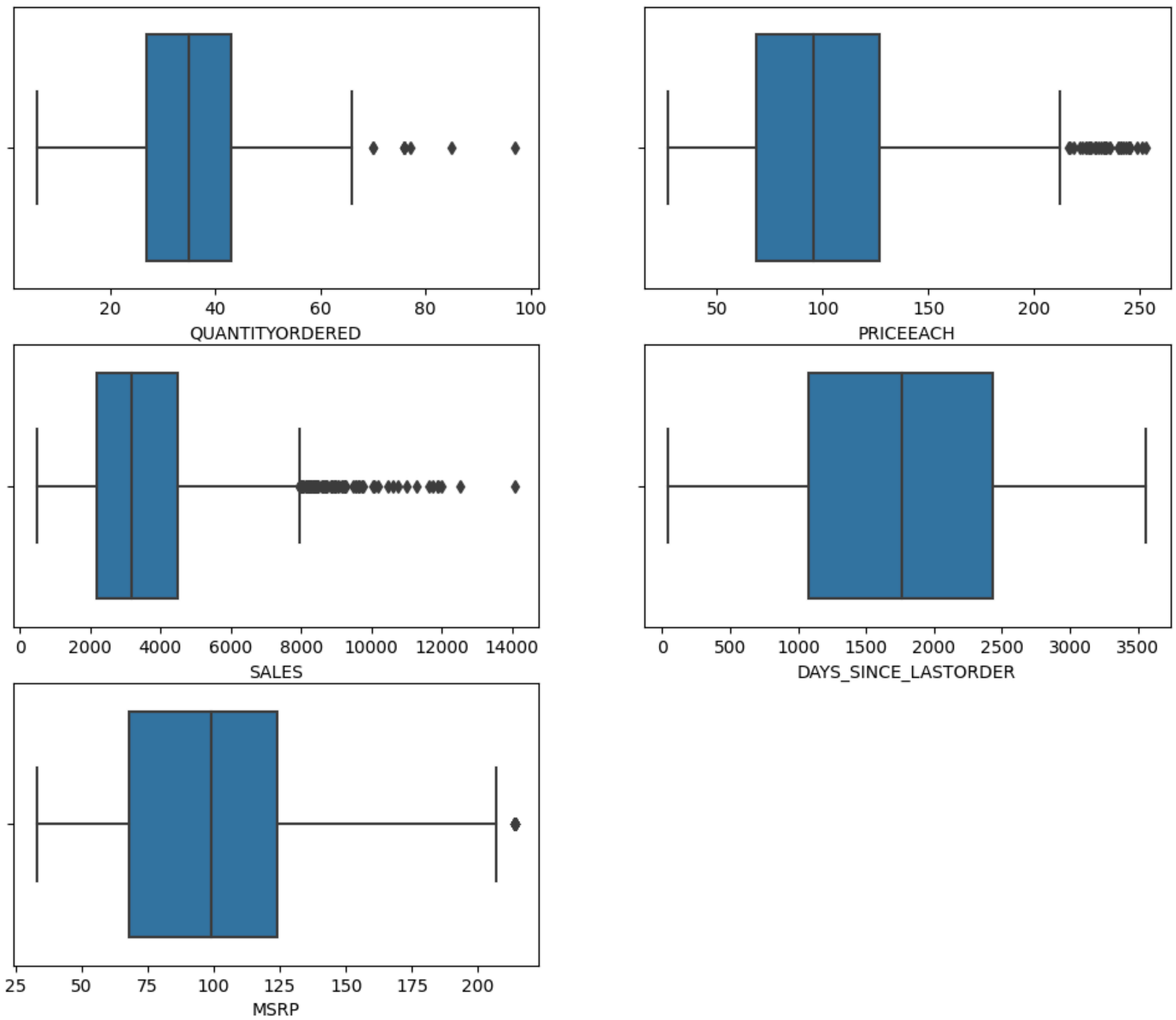
- No distribution (QUANTITYORDERED, PRICEEACH, SALES, DAYS_SINCE_LASTORDER and MSRP) is evenly distributed (symmetric).

In [19]:

```
# Box Plots for QUANTITYORDERED, PRICEEACH, SALES, DAYS_SINCE_LASTORDER and MSRP

fig, axes = plt.subplots(3,2, figsize=(12, 10))

sns.boxplot(ax=axes[0, 0], data=df_sales, x='QUANTITYORDERED')
sns.boxplot(ax=axes[0, 1], data=df_sales, x='PRICEEACH')
sns.boxplot(ax=axes[1, 0], data=df_sales, x='SALES')
sns.boxplot(ax=axes[1, 1], data=df_sales, x='DAYS_SINCE_LASTORDER')
sns.boxplot(ax=axes[2, 0], data=df_sales, x='MSRP')
axes[2,1].axis("off")

axes[0, 0].set(xlabel='QUANTITYORDERED')
axes[0, 1].set(xlabel='PRICEEACH')
axes[1, 0].set(xlabel='SALES')
axes[1, 1].set(xlabel='DAYS_SINCE_LASTORDER')
axes[2, 0].set(xlabel='MSRP')

plt.suptitle('Fig 7: Box Plots: QUANTITYORDERED, PRICEEACH, SALES, DAYS_SINCE_LASTO
```

```
plt.show()
```

Fig 7: Box Plots: QUANTITYORDERED, PRICEEACH, SALES, DAYS_SINCE_LASTORDER, MSRP



Observations and Insights:

- Except DAYS_SINCE_LASTORDER column other columns have few outliers.

Bivariate Analysis

Correlation among variables

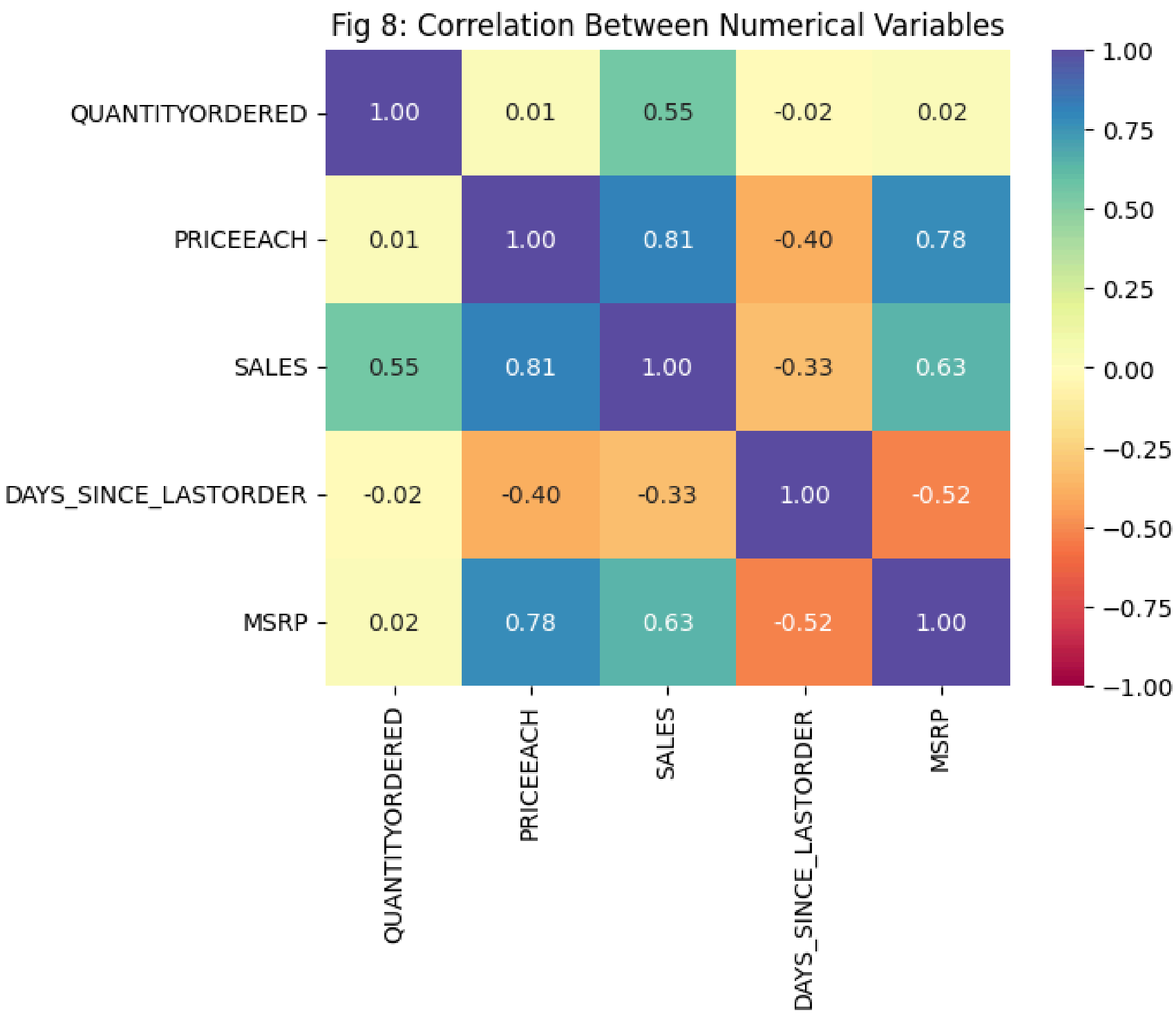
In [20]:
```
# orrelation between numerical variables in the dataset

col_list = ['QUANTITYORDERED','PRICEEACH','SALES','DAYS_SINCE_LASTORDER','MSRP']
corr = df_sales[col_list].corr()
corr
```

Out[20]:

	QUANTITYORDERED	PRICEEACH	SALES	DAYS_SINCE_LASTORD
QUANTITYORDERED	1.000000	0.010161	0.553359	-0.021
PRICEEACH	0.010161	1.000000	0.808287	-0.397
SALES	0.553359	0.808287	1.000000	-0.334
DAYS_SINCE_LASTORDER	-0.021923	-0.397092	-0.334274	1.000
MSRP	0.020551	0.778393	0.634849	-0.524

In [21]:

```
# Heatmap to plot correlation between numerical variables in the dataset

col_list = ['QUANTITYORDERED','PRICEEACH','SALES','DAYS_SINCE_LASTORDER','MSRP']
sns.heatmap(df_sales[col_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap
plt.title('Fig 8: Correlation Between Numerical Variables')
plt.show()
```



Observations and Insights:

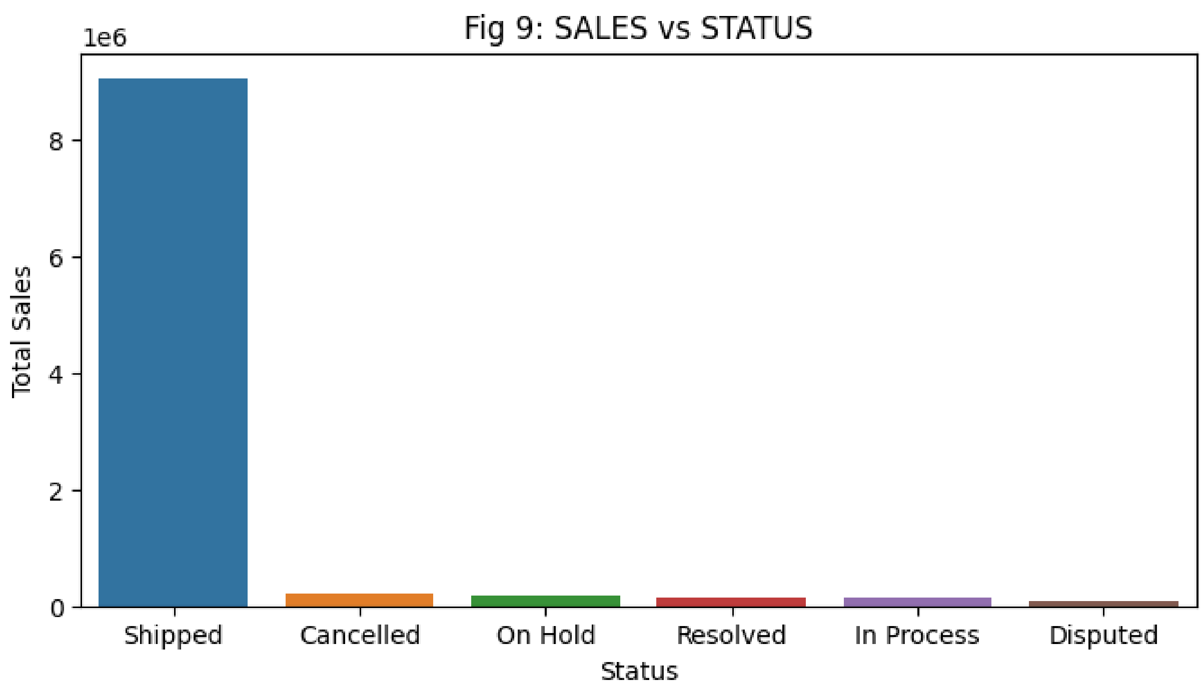
1. There is moderate correlation between QUANTITYORDERED and SALES.
2. There is moderate correlation between PRICEEACH and MSRP.

3. There is moderate correlation between SALES and PRICEEACH.
4. There is moderate correlation between MSRP and SALES.

Multivariate Analysis

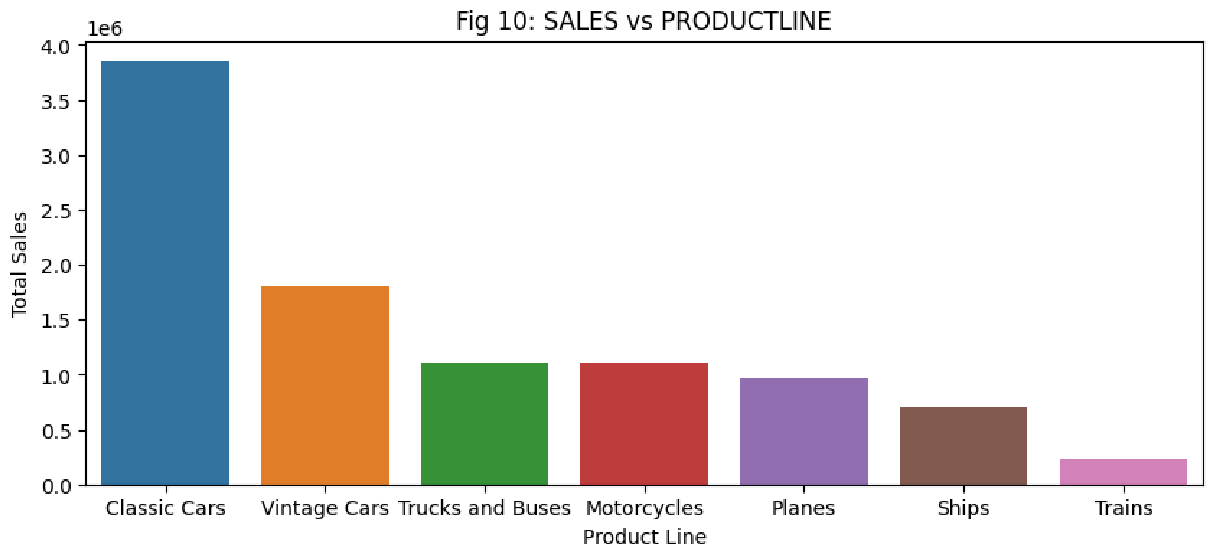
SALES vs STATUS

In [22]:
```
# Bar Plot for SALES vs STATUS

df_ss = df_sales.groupby(['STATUS']).agg(Total_Sales=('SALES','sum')).sort_values(b

plt.figure(figsize=(8,4))
sns.barplot(data=df_ss, x='STATUS', y='Total_Sales')
plt.title('Fig 9: SALES vs STATUS')
plt.xlabel('Status')
plt.ylabel('Total Sales')
plt.show()
```



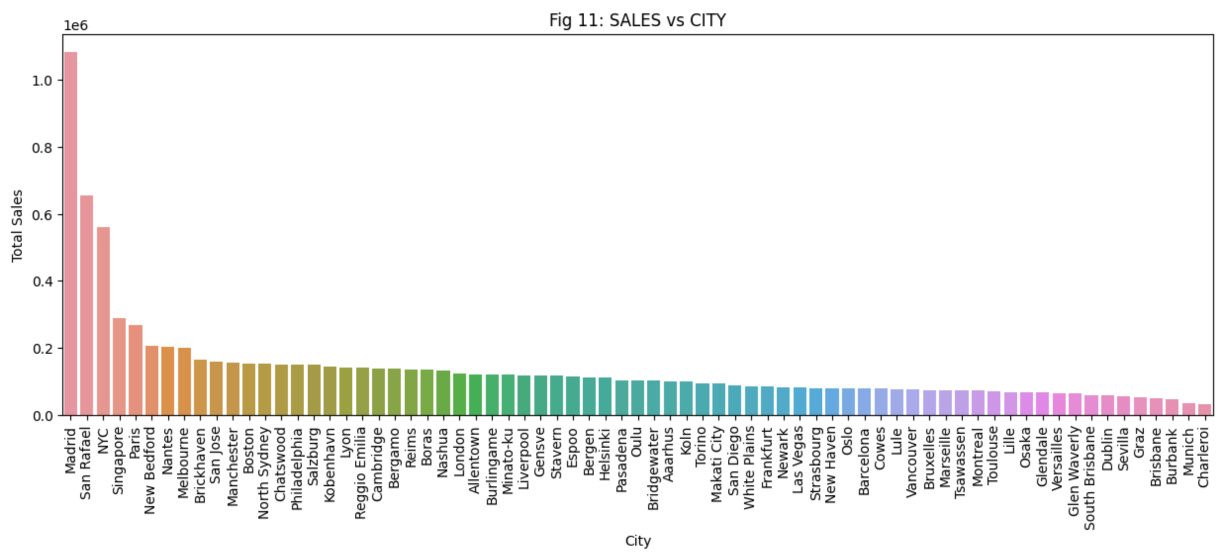
SALES vs PRODUCTLINE

In [23]:
```
# Bar Plot for SALES vs PRODUCTLINE

df_sp = df_sales.groupby(['PRODUCTLINE']).agg(Total_Sales=('SALES','sum')).sort_val

plt.figure(figsize=(10,4))
sns.barplot(data=df_sp, x='PRODUCTLINE', y='Total_Sales')
plt.title('Fig 10: SALES vs PRODUCTLINE')
plt.xlabel('Product Line')
plt.ylabel('Total Sales')
plt.show()
```



SALES vs CITY

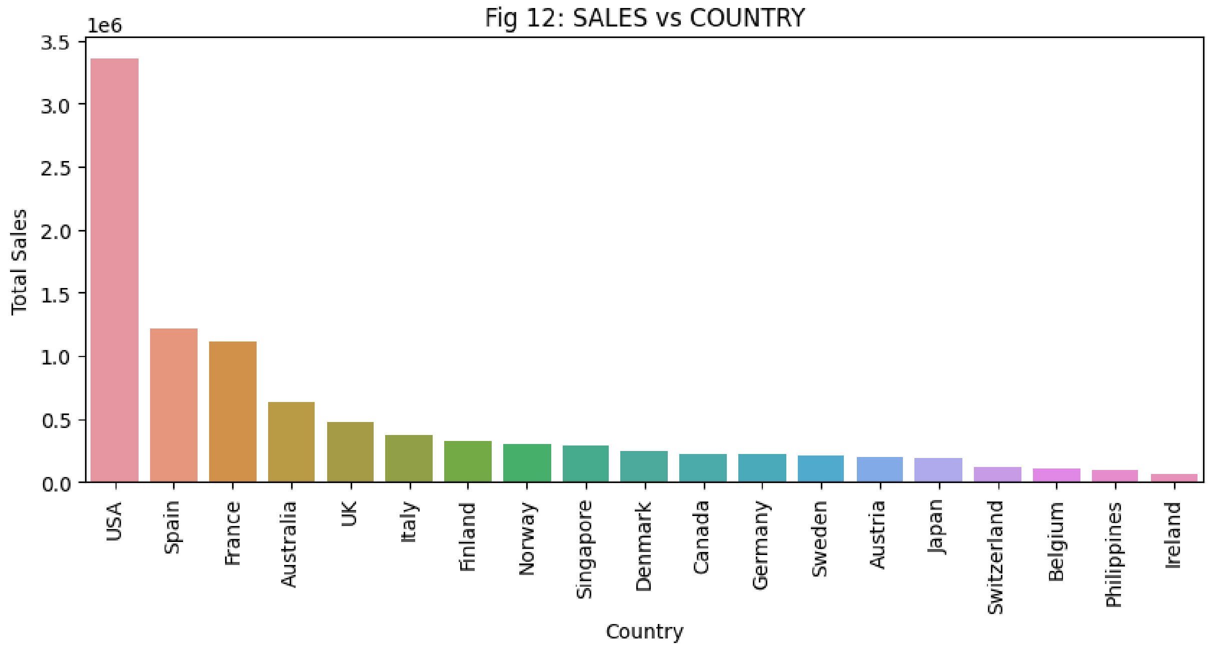
In [24]:
```
# Bar Plot for SALES vs CITY

df_sc = df_sales.groupby(['CITY']).agg(Total_Sales=('SALES','sum')).sort_values(by=

plt.figure(figsize=(15,5))
sns.barplot(data=df_sc, x='CITY', y='Total_Sales')
plt.title('Fig 11: SALES vs CITY')
plt.xticks(rotation=90)
plt.xlabel('City')
plt.ylabel('Total Sales')
plt.show()
```



SALES vs COUNTRY

In [25]:
```
# Bar Plot for SALES vs COUNTRY

df_sco = df_sales.groupby(['COUNTRY']).agg(Total_Sales=('SALES','sum')).sort_values
```

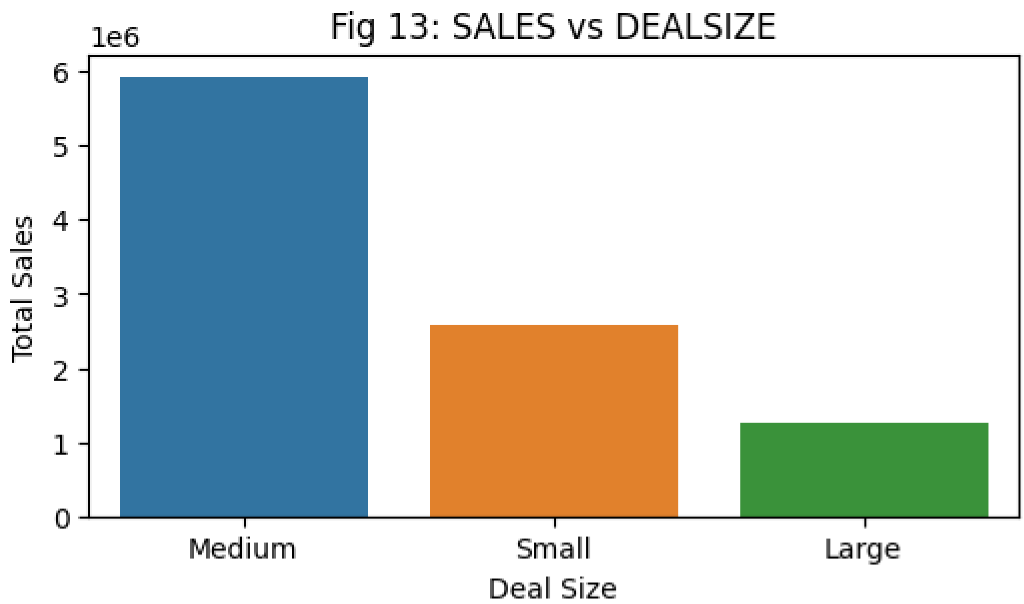
```
plt.figure(figsize=(10,4))
sns.barplot(data=df_sco, x='COUNTRY', y='Total_Sales')
plt.title('Fig 12: SALES vs COUNTRY')
plt.xticks(rotation=90)
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.show()
```



SALES vs DEALSIZE

In [26]:

```
# Bar Plot for SALES vs DEALSIZE

df_sd = df_sales.groupby(['DEALSIZE']).agg(Total_Sales=('SALES','sum')).sort_values

plt.figure(figsize=(6,3))
sns.barplot(data=df_sd, x='DEALSIZE', y='Total_Sales')
plt.title('Fig 13: SALES vs DEALSIZE')
plt.xlabel('Deal Size')
plt.ylabel('Total Sales')
plt.show()
```



Observations and Insights:

1. Maximum sales have Shipped as order status while Disputed is the lowest.
2. Maximum sales have Classic Cars as product line while Trains is the lowest.
3. Madrid city has highest sales while Charleroi city has lowest sales.
4. USA country has highest sales while Ireland country has lowest sales.
5. Maximum sales have Medium as deal size while Large is the lowest.

Sales Trend - Yearly, Quarterly, Monthly, Weekly, Daily

In [27]:
```
df_sales_ex = pd.read_excel("sales_data.xlsx",parse_dates=True,index_col=5)
df_sales_ts = df_sales_ex[['SALES']]
```

In [28]:
```
df_sales_ts.index.name = 'Time_Stamp' # Renaming index name
df_sales_ts.head()
```

Out[28]:

	SALES
Time_Stamp	
2018-02-24	2871.00
2018-05-07	2765.90
2018-07-01	3884.34
2018-08-25	3746.70
2018-10-28	3479.76

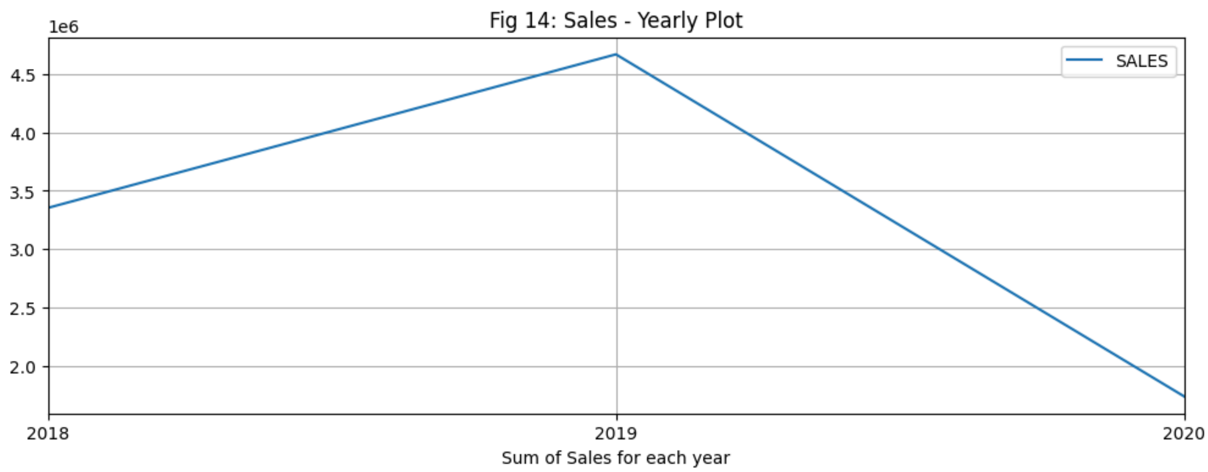
In [29]:
```
df_yearly_sum = df_sales_ts.resample('A').sum()
df_yearly_sum.head()
```

Out[29]:

	SALES
Time_Stamp	
2018-12-31	3353014.06
2019-12-31	4669924.56
2020-12-31	1737283.09

In [30]:

```
df_yearly_sum.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 14: Sales - Yearly Plot')
plt.xlabel('Sum of Sales for each year')
plt.show()
```



In [31]:

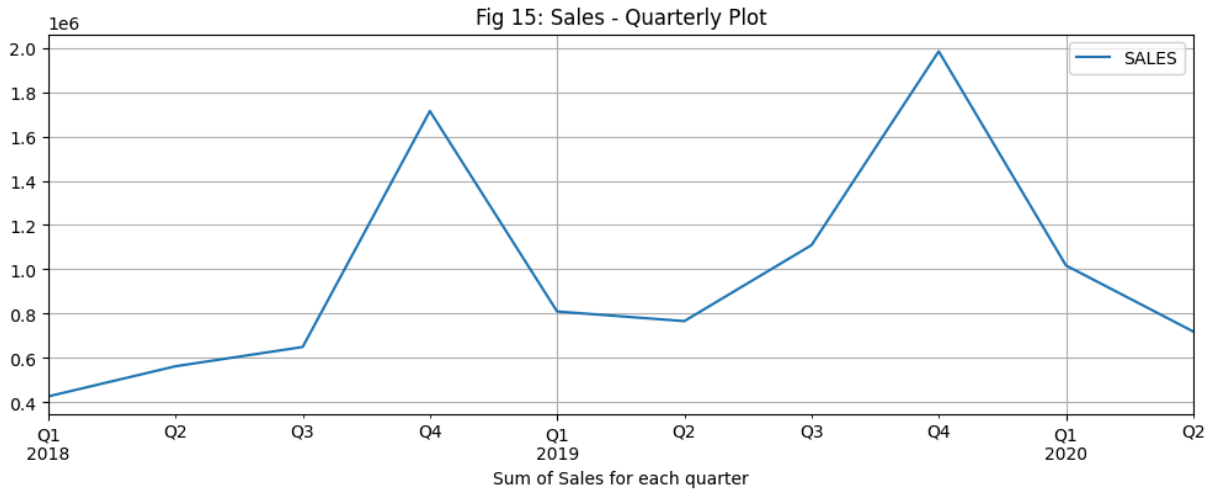
```
df_quarterly_sum = df_sales_ts.resample('Q').sum()
df_quarterly_sum.head()
```

Out[31]:

	SALES
Time_Stamp	
2018-03-31	426399.11
2018-06-30	562365.22
2018-09-30	649514.54
2018-12-31	1714735.19
2019-03-31	809841.36

In [32]:

```
df_quarterly_sum.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 15: Sales - Quarterly Plot')
plt.xlabel('Sum of Sales for each quarter')
plt.show()
```

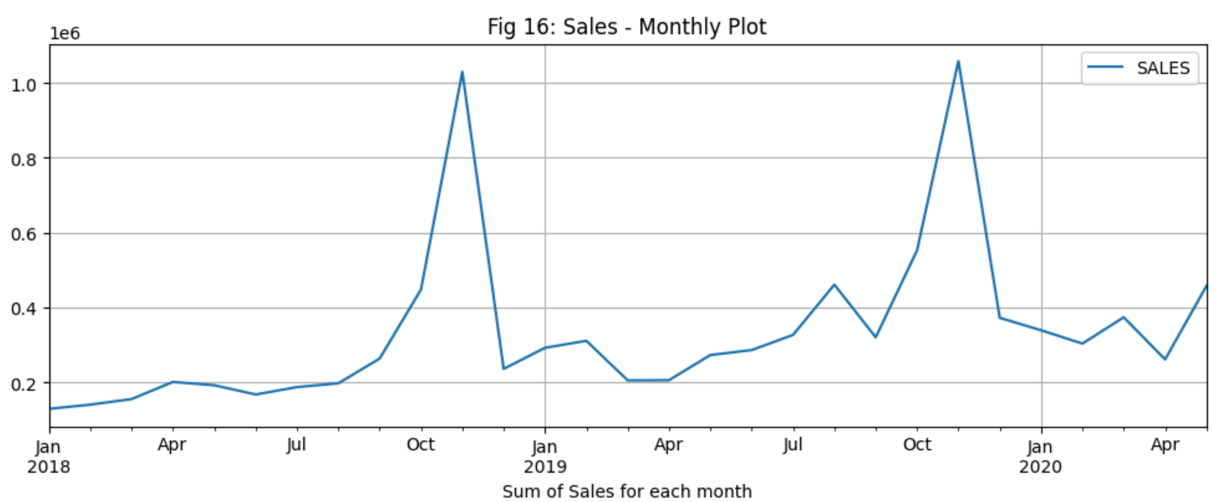


In [33]:
```
df_monthly_sum = df_sales_ts.resample('M').sum()
df_monthly_sum.head()
```

Out[33]:

	SALES
Time_Stamp	
2018-01-31	129753.60
2018-02-28	140836.19
2018-03-31	155809.32
2018-04-30	201609.55
2018-05-31	192673.11

In [34]:
```
df_monthly_sum.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 16: Sales - Monthly Plot')
plt.xlabel('Sum of Sales for each month')
plt.show()
```

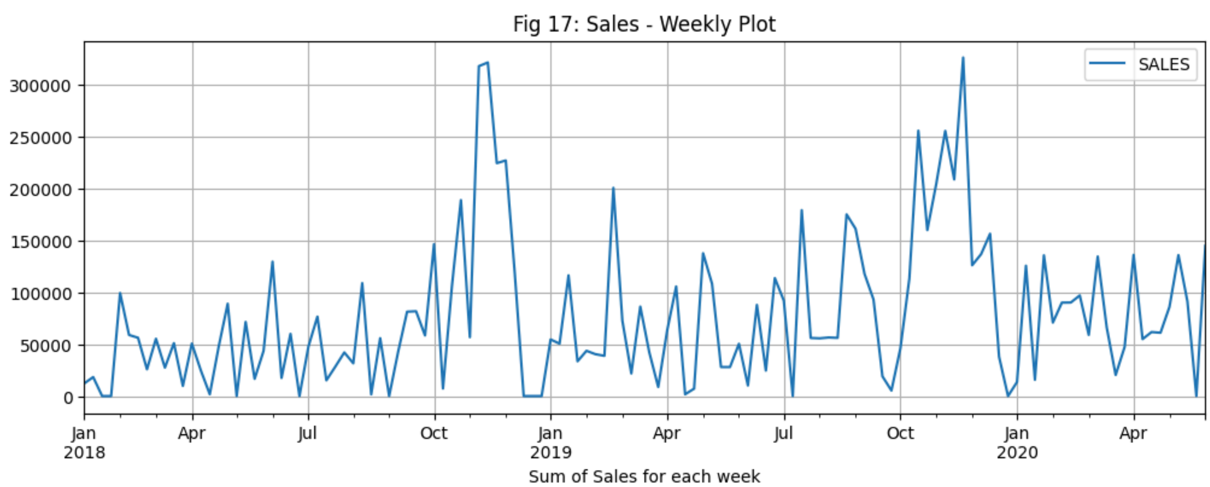


In [35]:
```
df_weekly_sum = df_sales_ts.resample('W').sum()
df_weekly_sum.head()
```

Out[35]:

Time_Stamp	SALES
2018-01-07	12133.25
2018-01-14	18296.39
2018-01-21	0.00
2018-01-28	0.00
2018-02-04	99323.96

In [36]:
```
df_weekly_sum.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 17: Sales - Weekly Plot')
plt.xlabel('Sum of Sales for each week')
plt.show()
```

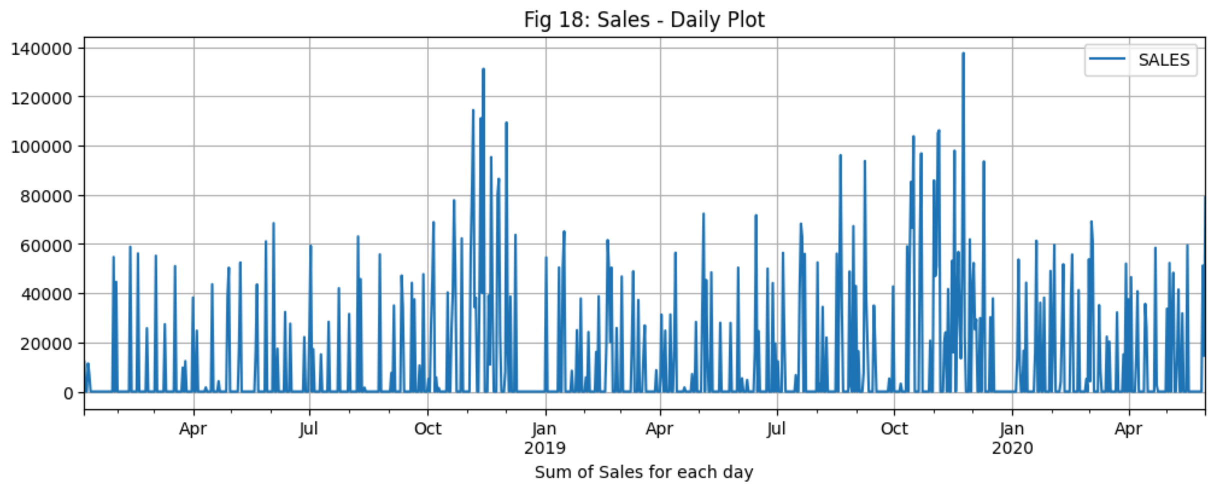


In [37]:
```
df_daily_sum = df_sales_ts.resample('D').sum()
df_daily_sum.head()
```

Out[37]:

Time_Stamp	SALES
2018-01-06	12133.25
2018-01-07	0.00
2018-01-08	0.00
2018-01-09	11432.34
2018-01-10	6864.05

In [38]:
```
df_daily_sum.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 18: Sales - Daily Plot')
plt.xlabel('Sum of Sales for each day')
plt.show()
```



Observations and Insights:

1. Trend is visible in Sales for each quarter, month, week and day of the years.
2. Sales is highest in the year 2019 followed by 2018 and 2020.
3. Sales is highest in the 4th quarter of each year.
4. Sales is highest in the November month of each year.
5. Sales is highest in the weeks of November and December months.
6. Sales is highest in the days of November and December months.

Part B

Problem Statement

A grocery store shared the transactional data with you. Your job is to conduct a thorough analysis of Point of Sale (POS) data, identify the most commonly occurring sets of items in the customer orders, and provide recommendations through which a grocery store can increase its revenue by popular combo offers & discounts for customers.

Understanding the structure of data

In [39]:
```
df_order = pd.read_csv('dataset_group.csv')

df_order.head() # Returns first 5 rows
```

Out[39]:

	Date	Order_id	Product
0	2018-01-01	1	yogurt
1	2018-01-01	1	pork
2	2018-01-01	1	sandwich bags
3	2018-01-01	1	lunch meat
4	2018-01-01	1	all- purpose

Number of rows and columns in the dataset

In [40]:
```
# checking shape of the data

rows = str(df_order.shape[0])
columns = str(df_order.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

```
There are 20641 rows and 3 columns in the dataset.
```

Datatypes of the different columns in the dataset

In [41]:
```
df_order.info() # Concise summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20641 entries, 0 to 20640
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Date      20641 non-null  object
 1   Order_id  20641 non-null  int64 
 2   Product   20641 non-null  object
dtypes: int64(1), object(2)
memory usage: 483.9+ KB
```

There are 3 columns in the dataset. Out of which 1 have integer data type and 2 have object data type.

Finding missing values in the dataset

In [42]:
```
df_order.isna().sum()  # Count NaN values in all columns of dataset
```

Out[42]:
```
Date        0
Order_id    0
Product     0
dtype: int64
```

There are no missing values in the dataset.

Checking for Duplicates

In [43]:
```
df_order.duplicated().sum() # Checking for duplicates
```

Out[43]:
```
4730
```

There are 4730 duplicate rows in the dataset.

Statistical summary of the data

In [44]:
```
# Summary statistics of the numerical data

df_order.describe().T
```

Out[44]:

	count	mean	std	min	25%	50%	75%	max
Order_id	20641.0	575.986289	328.557078	1.0	292.0	581.0	862.0	1139.0

Observations and Insights:

Minimum Order Id is 1 and maximum Order Id is 1139.

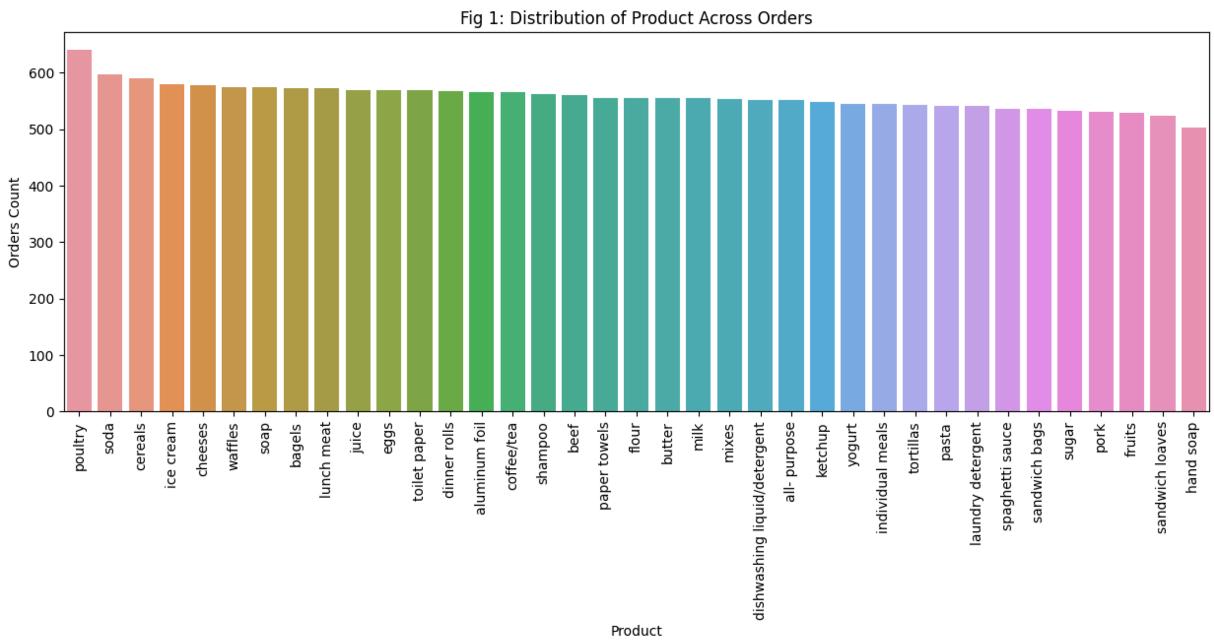
Exploratory Data Analysis (EDA)

Product

In [45]:
```
# Check unique Product

df_order['Product'].value_counts() # Frequency of each distinct value in the Produc
```

Out[45]:
```
Product
poultry                           640
soda                              597
cereals                           591
ice cream                         579
cheeses                           578
waffles                           575
soap                              574
bagels                            573
lunch meat                        573
juice                             570
eggs                              570
toilet paper                      569
dinner rolls                      567
aluminum foil                     566
coffee/tea                        565
shampoo                           562
beef                              561
paper towels                      556
flour                             555
butter                            555
milk                              555
mixes                             554
dishwashing liquid/detergent      551
all- purpose                      551
ketchup                           548
yogurt                            545
individual meals                  544
tortillas                         543
pasta                             542
laundry detergent                 542
spaghetti sauce                   536
sandwich bags                     536
sugar                             533
pork                              531
fruits                            529
sandwich loaves                   523
hand soap                         502
Name: count, dtype: int64
```

In [46]:
```
# Count Plot - Distribution of Product across orders

plt.figure(figsize=(15,5))
sns.countplot(data=df_order, x='Product', order = df_order['Product'].value_counts(
plt.title('Fig 1: Distribution of Product Across Orders')
plt.xticks(rotation=90)
plt.xlabel('Product')
plt.ylabel('Orders Count')
plt.show()
```



Observations and Insights:

Poultry product has the maximum orders while Hand Soap product has the lowest orders.

Product Trend - Yearly, Quarterly, Monthly, Weekly, Daily

In [47]:
```
df_orders_ex = pd.read_csv("dataset_group.csv",parse_dates=True,index_col=0)
df_orders_ts = df_orders_ex[['Product']]
```

In [48]:
```
df_orders_ts.index.name = 'Time_Stamp' # Renaming index name
df_orders_ts.head()
```

Out[48]:

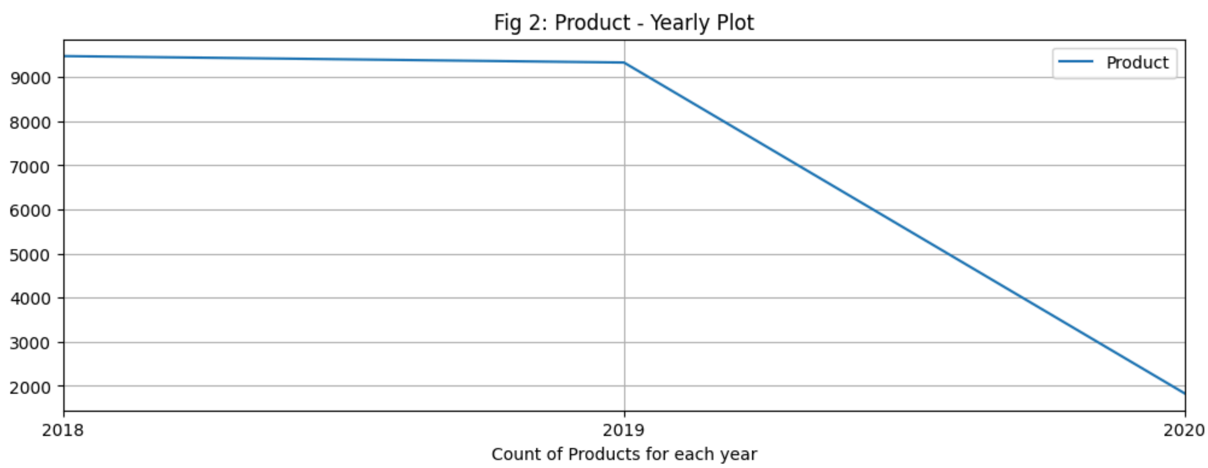
	Product
Time_Stamp	
2018-01-01	yogurt
2018-01-01	pork
2018-01-01	sandwich bags
2018-01-01	lunch meat
2018-01-01	all- purpose

In [49]:
```
df_yearly_count = df_orders_ts.resample('A').count()
df_yearly_count.head()
```

Out[49]:

Time_Stamp	Product
2018-12-31	9479
2019-12-31	9333
2020-12-31	1829

In [50]:
```
df_yearly_count.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 2: Product - Yearly Plot')
plt.xlabel('Count of Products for each year')
plt.show()
```

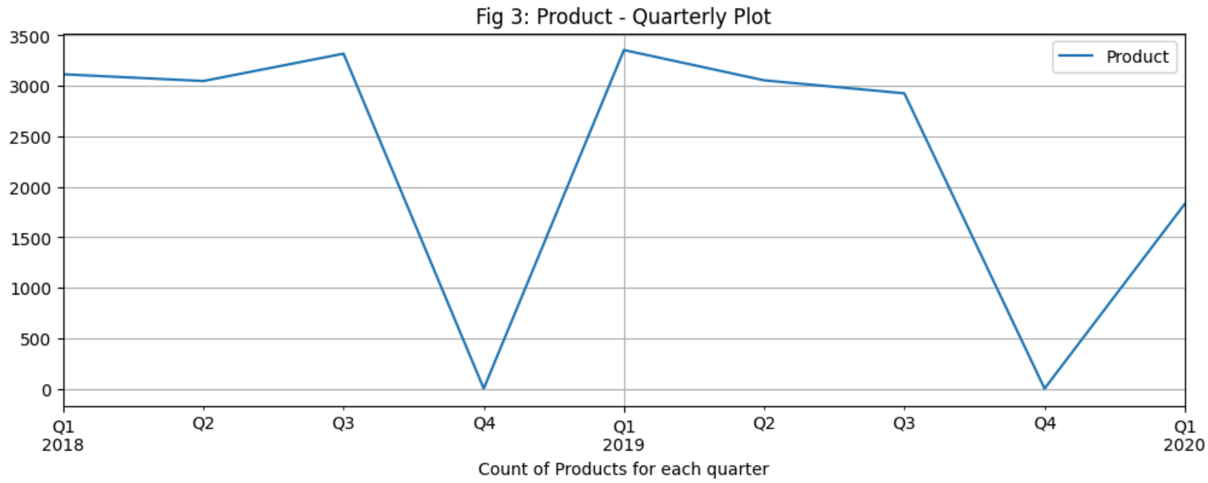


In [51]:
```
df_quarterly_count = df_orders_ts.resample('Q').count()
df_quarterly_count.head()
```

Out[51]:

Time_Stamp	Product
2018-03-31	3114
2018-06-30	3047
2018-09-30	3318
2018-12-31	0
2019-03-31	3354

In [52]:
```
df_quarterly_count.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 3: Product - Quarterly Plot')
plt.xlabel('Count of Products for each quarter')
plt.show()
```

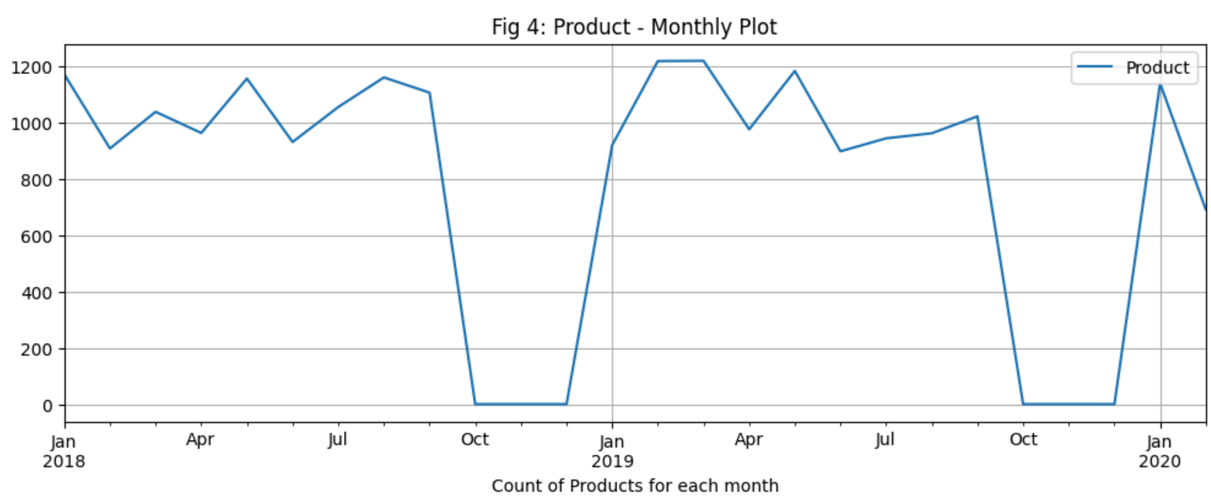


In [53]:
```
df_monthly_count = df_orders_ts.resample('M').count()
df_monthly_count.head()
```

Out[53]:

	Product
Time_Stamp	
2018-01-31	1170
2018-02-28	907
2018-03-31	1037
2018-04-30	962
2018-05-31	1155

In [54]:
```
df_monthly_count.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 4: Product - Monthly Plot')
plt.xlabel('Count of Products for each month')
plt.show()
```



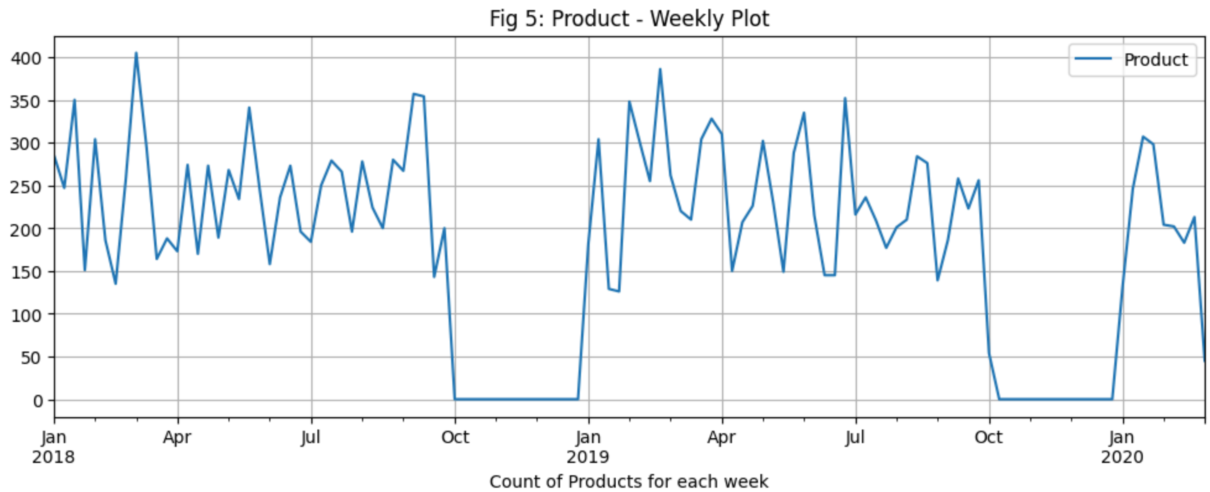
In [55]:
```
df_weekly_count = df_orders_ts.resample('W').count()
df_weekly_count.head()
```

Out[55]:

	Product
Time_Stamp	
2018-01-07	285
2018-01-14	247
2018-01-21	350
2018-01-28	151
2018-02-04	304

In [56]:

```
df_weekly_count.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 5: Product - Weekly Plot')
plt.xlabel('Count of Products for each week')
plt.show()
```



In [57]:

```
df_daily_count = df_orders_ts.resample('D').count()
df_daily_count.head()
```

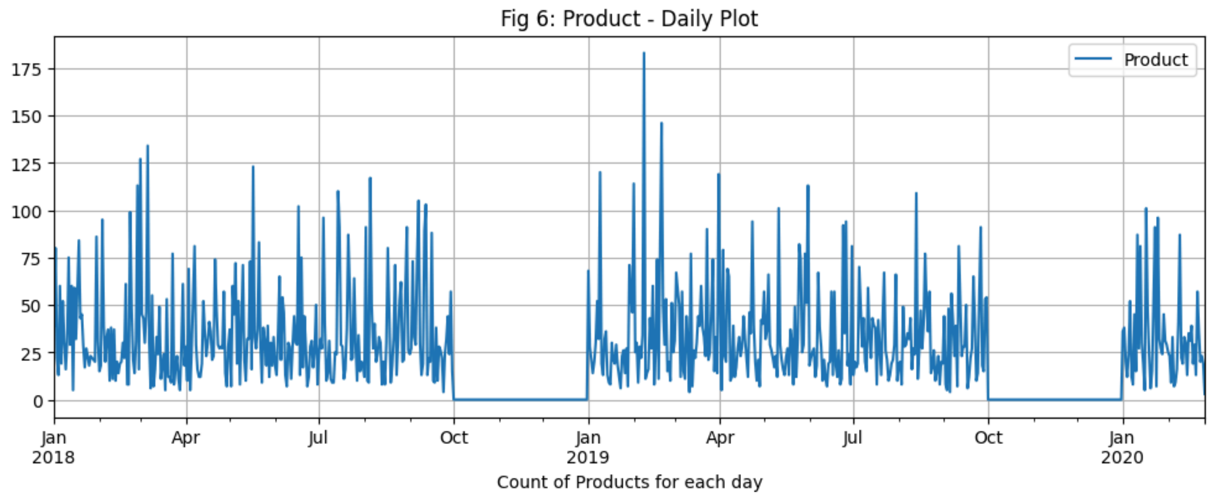
Out[57]:

	Product
Time_Stamp	
2018-01-01	39
2018-01-02	80
2018-01-03	22
2018-01-04	13
2018-01-05	60

In [58]:

```
df_daily_count.plot(figsize=(12,4))
plt.grid()
plt.title('Fig 6: Product - Daily Plot')
```

```python
plt.xlabel('Count of Products for each day')
plt.show()
```



Observations and Insights:

1. Number of products sold is highest in the year 2018 followed by 2019 and 2020.
2. Number of products sold shows trend in the 1st, 2nd and 3rd quarter of each year. However it is 0 in the 4th quarter of each year.
3. Number of products sold shows trend from January to September months of each year. However it is 0 from October to December months of each year.
4. Number of products sold shows trend in the weeks from January to September months of each year. However it is 0 in the weeks from October to December months of each year.
5. Number of products sold shows trend in the days from January to September months of each year. However it is 0 in the days from October to December months of each year.