Install necessary modules and libraries

```
!pip install openpyxl
```

```
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl.metadata (2.5 kB)
Collecting et-xmlfile (from openpyxl)
  Downloading et_xmlfile-2.0.0-py3-none-any.whl.metadata (2.7 kB)
Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 250.9/250.9 kB 2.2 MB/s eta 0:00:00
Downloading et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-2.0.0 openpyxl-3.1.5
```

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

Load Dataset

```
!wget https://archive.ics.uci.edu/static/public/352/online+retail.zip
```

```
--2025-04-17 01:01:23--  https://archive.ics.uci.edu/static/public/352/online+retail.zip
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified
Saving to: ‘online+retail.zip’

online+retail.zip       [    <=>             ]  22.62M  29.1MB/s    in 0.8s

2025-04-17 01:01:24 (29.1 MB/s) - ‘online+retail.zip’ saved [23715478]
```

```
!unzip online+retail.zip
```

```
Archive:  online+retail.zip
 extracting: Online Retail.xlsx
```

```
import time
stime = time.time()
```

```
df1 = pd.read_excel("Online Retail.xlsx", dtype={'InvoiceNo': 'string', 'StockCode': 'string', 'Description': 'string', 'Country':
df1.head(3)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

```
df1.shape
```

```
(541909, 8)
```

```
df1[df1.InvoiceNo=="C536379"]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
141	C536379	D	Discount	-1	2010-12-01 09:41:00	27.5	14527.0	United Kingdom

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  string
 1   StockCode    541909 non-null  string
 2   Description  540455 non-null  string
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
```

```
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  string
dtypes: datetime64[ns](1), float64(2), int64(1), string(4)
memory usage: 33.1 MB
```

Data Cleaning: Handle Missing Values

```
df1.isnull().sum()
```

	0
InvoiceNo	0
StockCode	0
Description	1454
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0

dtype: int64

```
df1[df1['Description'].isnull()].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	<NA>	56	2010-12-01 11:52:00	0.0	NaN	United Kingdom
1970	536545	21134	<NA>	1	2010-12-01 14:32:00	0.0	NaN	United Kingdom
1971	536546	22145	<NA>	1	2010-12-01 14:33:00	0.0	NaN	United Kingdom
1972	536547	37509	<NA>	1	2010-12-01 14:33:00	0.0	NaN	United Kingdom
1987	536549	85226A	<NA>	1	2010-12-01 14:34:00	0.0	NaN	United Kingdom

```
df1[df1.StockCode=="22139"]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
106	536381	22139	RETROSPOT TEA SET CERAMIC 11 PC	23	2010-12-01 09:41:00	4.25	15311.0	United Kingdom
622	536414	22139	<NA>	56	2010-12-01 11:52:00	0.00	NaN	United Kingdom
6392	536942	22139	amazon	15	2010-12-03 12:08:00	0.00	NaN	United Kingdom
6885	536982	22139	RETROSPOT TEA SET CERAMIC 11 PC	10	2010-12-03 14:27:00	11.02	NaN	United Kingdom
7203	537011	22139	<NA>	-5	2010-12-03 15:38:00	0.00	NaN	United Kingdom
...	...	...	...	...	...	...	...	...
538411	581405	22139	RETROSPOT TEA SET CERAMIC 11 PC	1	2011-12-08 13:50:00	4.95	13521.0	United Kingdom
539531	581439	22139	RETROSPOT TEA SET CERAMIC 11 PC	1	2011-12-08 16:30:00	10.79	NaN	United Kingdom

```
df1[df1.StockCode=="22139"].Description.mode()
```

	Description
0	RETROSPOT TEA SET CERAMIC 11 PC

dtype: string

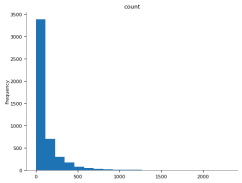
```
most_freq = df1[["StockCode","Description"]].value_counts().reset_index()
most_freq
```

	StockCode	Description	count
0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2302
1	22423	REGENCY CAKESTAND 3 TIER	2200
2	85099B	JUMBO BAG RED RETROSPOT	2159
3	47566	PARTY BUNTING	1727
4	20725	LUNCH BAG RED RETROSPOT	1638
...	...	...	...
4787	35833P	check	1
4788	21410	COUNTRY COTTAGE DOORSTOP GREEN	1
4789	21412	VINTAGE GOLD TINSEL REEL	1
4790	21414	SCALLOP SHELL SOAP DISH	1
4791	35972	check	1

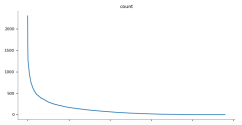
4792 rows × 3 columns

```
WARNING:root:Quickchart encountered unexpected dtypes in columns: "(['StockCode', 'Description'],)"
```

Distributions



Values



```
most_freq[most_freq.StockCode=="85123A"]
```

	StockCode	Description	count
0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2302
3300	85123A	CREAM HANGING HEART T-LIGHT HOLDER	9
4620	85123A	wrongly marked carton 22804	1
4658	85123A	?	1

```
most_freq = most_freq.groupby("StockCode").head(1)
most_freq.head(5)
```

	StockCode	Description	count
0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2302
1	22423	REGENCY CAKESTAND 3 TIER	2200
2	85099B	JUMBO BAG RED RETROSPOT	2159
3	47566	PARTY BUNTING	1727
4	20725	LUNCH BAG RED RETROSPOT	1638

```
most_freq[most_freq.StockCode=="85123A"]
```

	StockCode	Description	count
0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2302

```
most_freq.columns = ["StockCode", "freq_Description", "count"]
most_freq.head(3)
```

	StockCode	freq_Description	count
0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2302
1	22423	REGENCY CAKESTAND 3 TIER	2200
2	85099B	JUMBO BAG RED RETROSPOT	2159

```
df2 = df1.merge(most_freq, on="StockCode", how="left")
df2.head(3)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	freq_Description	count
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	WHITE HANGING HEART T-LIGHT HOLDER	2302.0
			WHITE METAL		2010-12-01			United	WHITE METAL	

```
df2[df2['Description'].isnull()]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	freq_Description	count
622	536414	22139	<NA>	56	2010-12-01 11:52:00	0.0	NaN	United Kingdom	RETROSPOT TEA SET CERAMIC 11 PC	988.0
1970	536545	21134	<NA>	1	2010-12-01 14:32:00	0.0	NaN	United Kingdom	<NA>	NaN
1971	536546	22145	<NA>	1	2010-12-01 14:33:00	0.0	NaN	United Kingdom	CHRISTMAS CRAFT HEART STOCKING	1.0
1972	536547	37509	<NA>	1	2010-12-01 14:33:00	0.0	NaN	United Kingdom	NEW ENGLAND MUG W GIFT BOX	2.0
1987	536549	85226A	<NA>	1	2010-12-01 14:34:00	0.0	NaN	United Kingdom	<NA>	NaN
...	...	...	...	...	...	...	...	...	...	...
535322	581199	84581	<NA>	-2	2011-12-07 18:26:00	0.0	NaN	United Kingdom	DOG TOY WITH PINK CROCHET SKIRT	91.0
535326	581203	23406	<NA>	15	2011-12-07 18:31:00	0.0	NaN	United Kingdom	HOME SWEET HOME KEY HOLDER	114.0

```
df2['Description'] = df2['Description'].mask(df2['Description'].isnull(), df2['freq_Description'])
df2.isnull().sum()
```

	0
InvoiceNo	0
StockCode	0
Description	112
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0
freq_Description	112
count	112

dtype: int64

```
df2.dropna(subset=['Description'], inplace=True)
df2.isnull().sum()
```

	0
InvoiceNo	0
StockCode	0
Description	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	134968
Country	0
freq_Description	0
count	0

dtype: int64

```
df2.drop(columns = ["freq_Description", "count"], inplace=True)
```

Data Cleaning: Handle Invalid Values

```
df2.describe()
```

	Quantity	InvoiceDate	UnitPrice	CustomerID
count	541797.000000	541797	541797.000000	406829.000000
mean	9.555919	2011-07-04 14:06:48.671255296	4.612067	15287.690570
min	-80995.000000	2010-12-01 08:26:00	-11062.060000	12346.000000
25%	1.000000	2011-03-28 11:36:00	1.250000	13953.000000
50%	3.000000	2011-07-20 08:59:00	2.080000	15152.000000
75%	10.000000	2011-10-19 11:41:00	4.130000	16791.000000
max	80995.000000	2011-12-09 12:50:00	38970.000000	18287.000000
std	218.103428	NaN	96.769831	1713.600303

```
df2[df2.Quantity<=0].shape
```

```
(10527, 8)
```

```
df2[df2.UnitPrice<=0].shape
```

```
(2405, 8)
```

```
# Remove negative or zero quantities and prices
df3 = df2[(df2['Quantity'] > 0) & (df2['UnitPrice'] > 0)]
df3.describe()
```

	Quantity	InvoiceDate	UnitPrice	CustomerID
count	530104.000000	530104	530104.000000	397884.000000
mean	10.542037	2011-07-04 20:16:05.225087744	3.907625	15294.423453
min	1.000000	2010-12-01 08:26:00	0.001000	12346.000000
25%	1.000000	2011-03-28 12:22:00	1.250000	13969.000000
50%	3.000000	2011-07-20 12:58:00	2.080000	15159.000000
75%	10.000000	2011-10-19 12:39:00	4.130000	16795.000000
max	80995.000000	2011-12-09 12:50:00	13541.330000	18287.000000
std	155.524124	NaN	35.915681	1713.141560

```
df3.Quantity.quantile(0.9999)
```

```
np.float64(1439.8763999990188)
```

Feature Engineering: Create New Columns

```
df3.head(2)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom

```
df4 = df3.copy()

df4['TotalSales'] = df4['Quantity'] * df4['UnitPrice']
df4.head(2)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSales
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30

```
df4['Month'] = df4['InvoiceDate'].dt.month
df4.sample(2)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSales	Month
472449	576688	23311	VINTAGE CHRISTMAS STOCKING	6	2011-11-16 12:16:00	2.55	16717.0	United Kingdom	15.3	11

```
df4['Month'] = df4['InvoiceDate'].dt.month
df4.sample(2)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSales	Month
368064	568938	84992	72 SWEETHEART FAIRY CAKE CASES	3	2011-09-29 14:46:00	0.55	17220.0	United Kingdom	1.65	9

Data Visualization and Insights

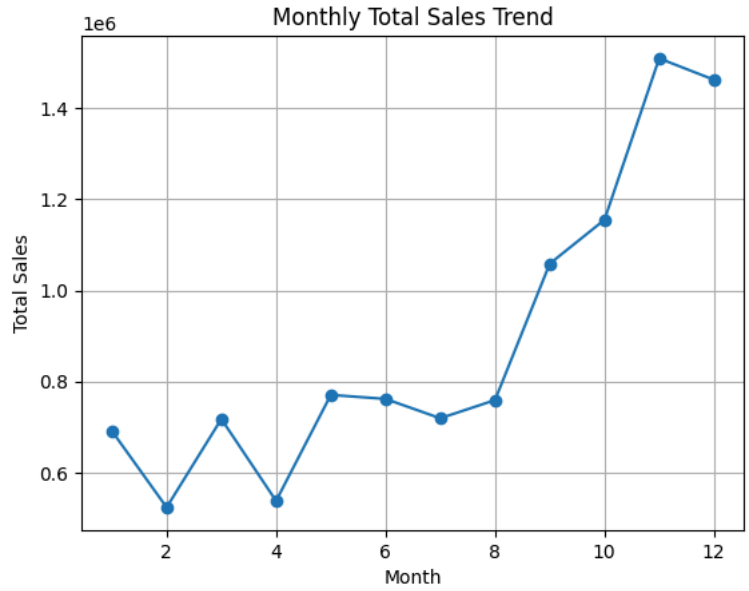
1. Plot Monthly Total Sales Trend

```
monthly_sales = df4.groupby('Month')['TotalSales'].sum()
monthly_sales
```

Month	TotalSales
1	691364.560
2	523631.890
3	717639.360
4	537808.621
5	770536.020
6	761739.900
7	719221.191
8	759138.380
9	1058590.172
10	1154979.300
11	1509496.330
12	1462538.820

dtype: float64

```
monthly_sales.plot(kind='line', title='Monthly Total Sales Trend', marker='o')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```



Insights

Total sales started rising up in August having a peek in November. This is likely due to the holiday season at the end of the year

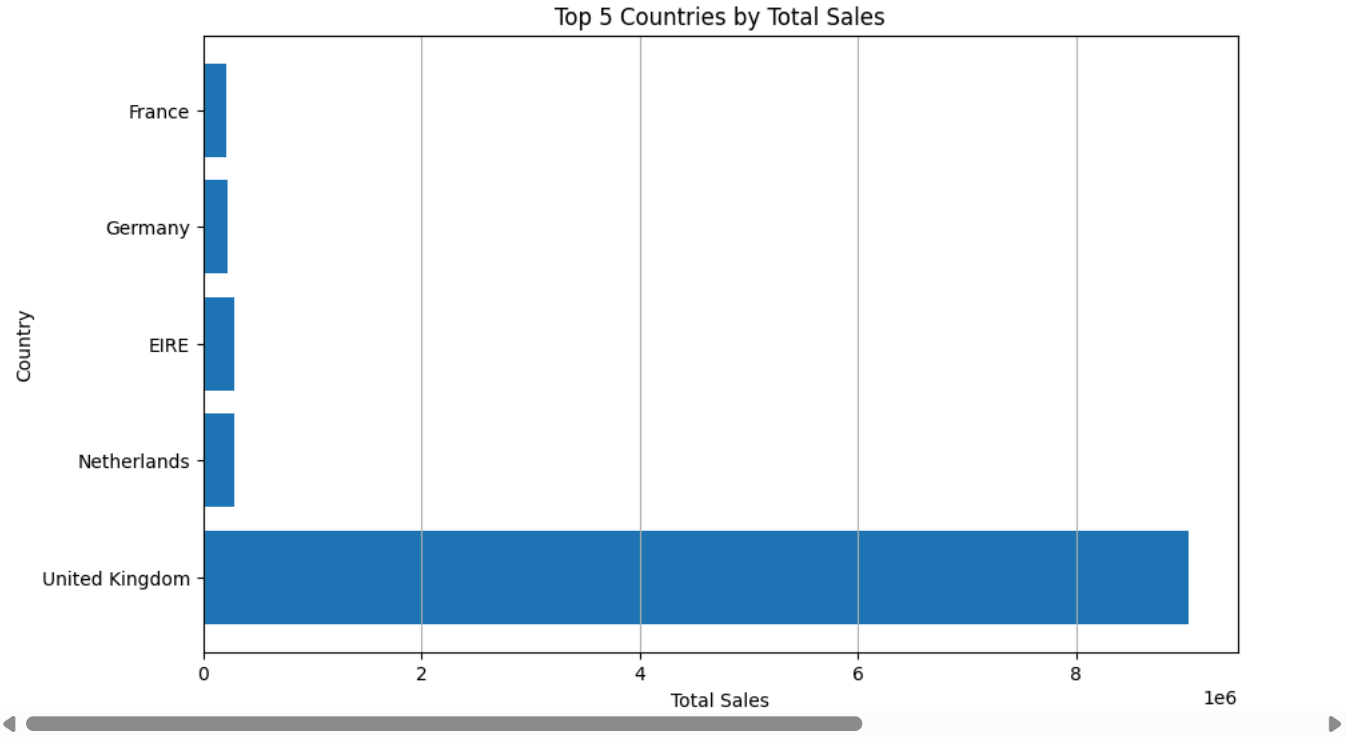
2. Top 5 countries based on total sales

```
df4.groupby('Country')['TotalSales'].sum().sort_values(ascending=False).head(5)
```

Country	TotalSales
United Kingdom	9025222.084
Netherlands	285446.340
EIRE	283453.960
Germany	228867.140
France	209715.110

dtype: float64

```
# prompt: plot horizontal bar chart for country wise monthly sales for top 5 countries

import matplotlib.pyplot as plt
top_5_countries = df4.groupby('Country')['TotalSales'].sum().sort_values(ascending=False).head(5)

plt.figure(figsize=(10, 6))
plt.barh(top_5_countries.index, top_5_countries.values)
plt.xlabel('Total Sales')
plt.ylabel('Country')
plt.title('Top 5 Countries by Total Sales')
plt.grid(axis='x')
plt.show()
```

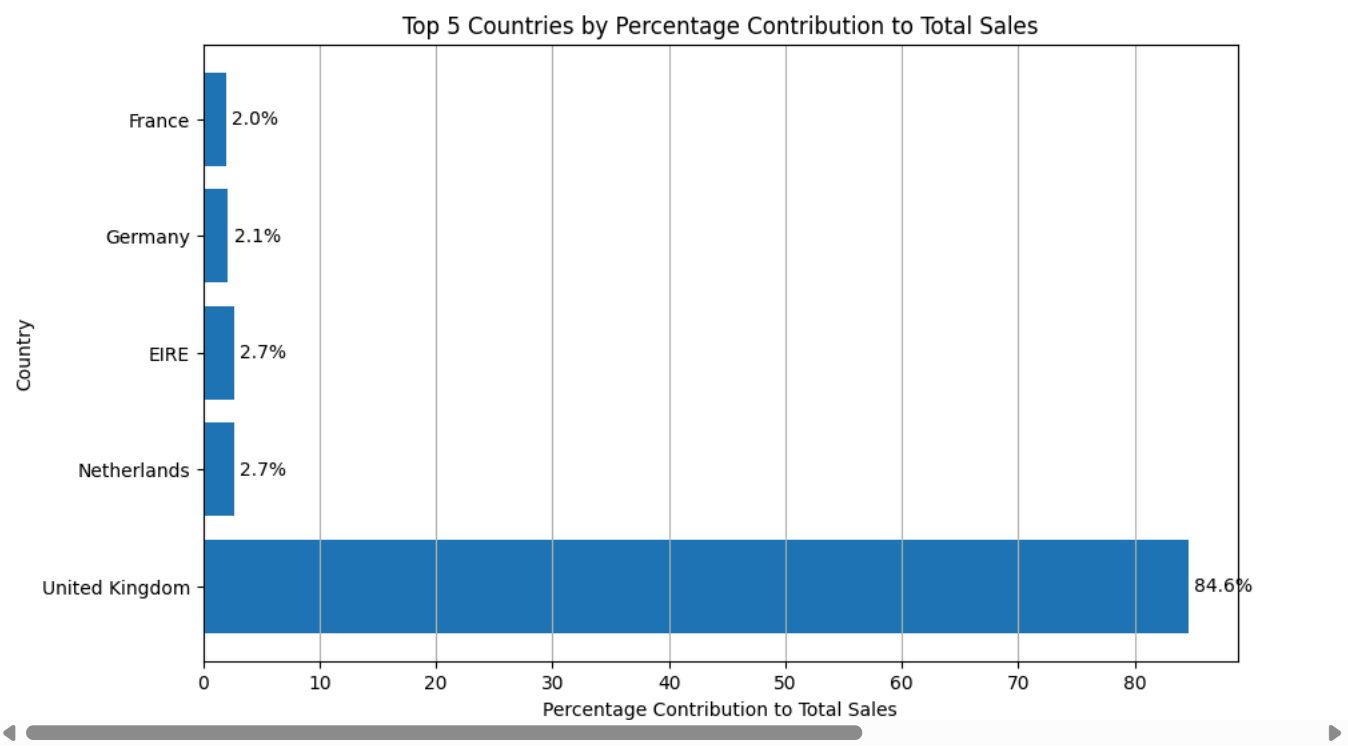


Insights

1. UK has the highest sales (around 9 million)
2. Netherlands, EIRE, Germany and France are the next 4 countries each having a sales of more than 2 million

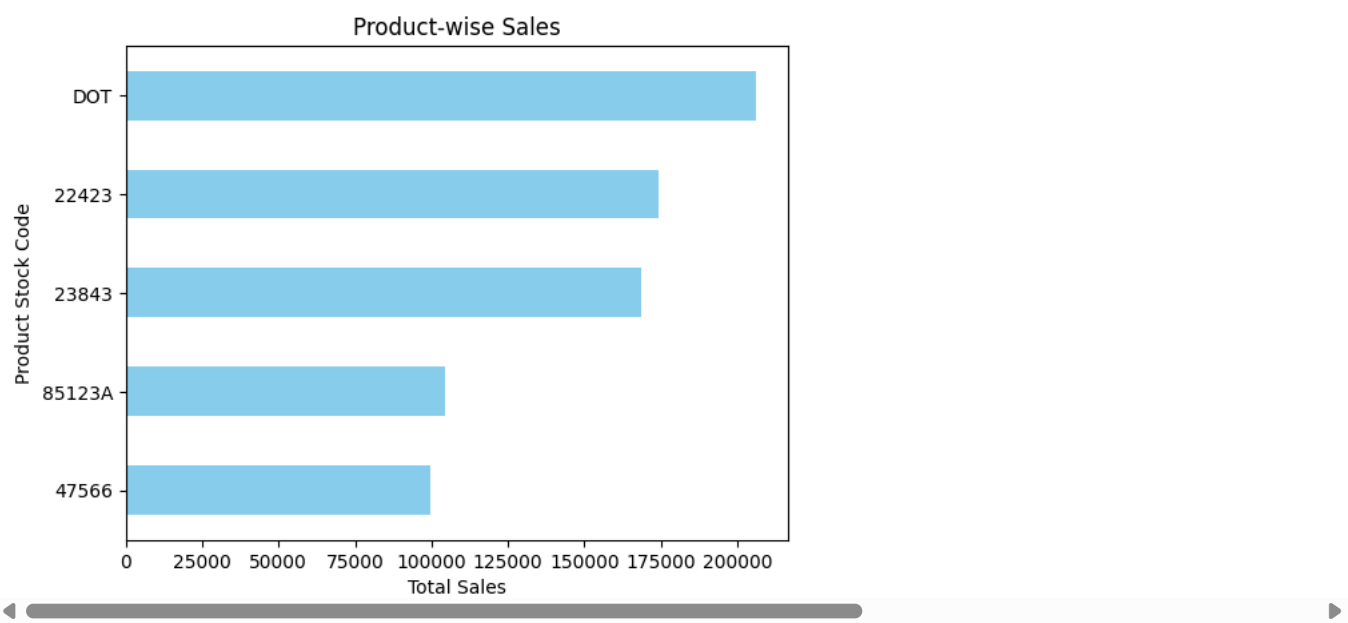
Since these countries cover the major sales revenues, we need to pay special attention to customers in these countries and make sure our product quality and service are the best. Also to break dependancy of sales from a single country we can focus on expanding sales in other countries as well

```
# prompt: Plot same chart as above but this time use percentage contribution. Show % on the bar

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


country_wise_sales = df4.groupby('Country')['TotalSales'].sum()
total_sales = country_wise_sales.sum()

top_5_countries = country_wise_sales.sort_values(ascending=False).head(5)
percentages = (top_5_countries / total_sales) * 100

plt.figure(figsize=(10, 6))
bars = plt.barh(top_5_countries.index, percentages)
plt.xlabel('Percentage Contribution to Total Sales')
plt.ylabel('Country')
plt.title('Top 5 Countries by Percentage Contribution to Total Sales')
plt.grid(axis='x')

# Add percentage labels to the bars
for bar, percentage in zip(bars, percentages):
    plt.text(bar.get_width() + 0.5, bar.get_y() + bar.get_height()/2, f'{percentage:.1f}%', va='center')

plt.show()
```



3. Top 5 products based on total sales

```
product_wise_sales = df4.groupby('StockCode')['TotalSales'].sum()

top_5_products = product_wise_sales.sort_values(ascending=False).head(5)
top_5_products.plot(kind='barh', color='skyblue')
plt.title('Product-wise Sales')
plt.xlabel('Total Sales')
plt.ylabel('Product Stock Code')
plt.gca().invert_yaxis()  # To show the highest sales at the top
plt.show()
```




```
product_wise_sales.sort_values(ascending=False)
```

StockCode	TotalSales
DOT	206248.770
22423	174484.740
23843	168469.600
85123A	104518.800
47566	99504.330
...	...
90084	0.850
21268	0.840
51014c	0.830
84227	0.420
PADS	0.003

3922 rows × 1 columns

dtype: float64

```
product_wise_sales.sum()
```

```
np.float64(10666684.544)
```

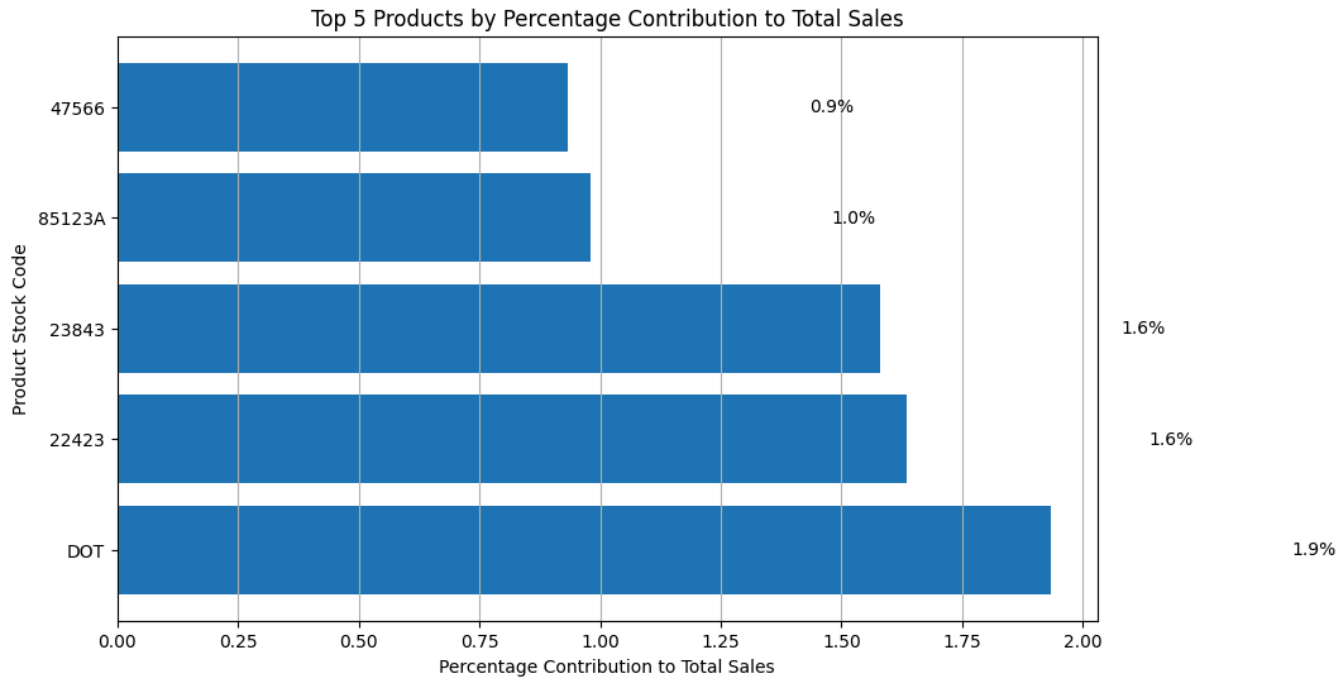
```
df4[df4.StockCode=="DOT"].Description.iloc[0]
```

```
'DOTCOM POSTAGE'
```

```
for stock_code in top_5_products.index:
    description = df4[df4.StockCode==stock_code].Description.iloc[0]
    print(f"{stock_code} ==> {description}")
```

```
DOT ==> DOTCOM POSTAGE
22423 ==> REGENCY CAKESTAND 3 TIER
23843 ==> PAPER CRAFT , LITTLE BIRDIE
85123A ==> WHITE HANGING HEART T-LIGHT HOLDER
47566 ==> PARTY BUNTING
```

```
# prompt: Plot same chart as above for product sales but use percentage this time. Show % on the bar

# Assuming df4 is already created from the previous code
total_sales = product_wise_sales.sum()
percentages = (top_5_products / total_sales) * 100

plt.figure(figsize=(10, 6))
bars = plt.barh(top_5_products.index, percentages)
plt.xlabel('Percentage Contribution to Total Sales')
plt.ylabel('Product Stock Code')
plt.title('Top 5 Products by Percentage Contribution to Total Sales')
plt.grid(axis='x')

# Add percentage labels to the bars
for bar, percentage in zip(bars, percentages):
    plt.text(bar.get_width() + 0.5, bar.get_y() + bar.get_height()/2, f'{percentage:.1f}%', va='center')

plt.show()
```



4. RFM Analysis (Recency, Frequency, Monetary)

```
df4['InvoiceDate'].max()
```

```
Timestamp('2011-12-09 12:50:00')
```

```
current_date = df4['InvoiceDate'].max() + pd.Timedelta(days=1)
rfm = df4.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (current_date - x.max()).days,
    'InvoiceNo': 'count',
    'TotalSales': 'sum'
})
rfm.columns = ['Recency', 'Frequency', 'Monetary']
rfm.head()
```

CustomerID	Recency	Frequency	Monetary
12346.0	326	1	77183.60
12347.0	2	182	4310.00
12348.0	75	31	1797.24
12349.0	19	73	1757.55
12350.0	310	17	334.40

```
rfm.describe()
```

	Recency	Frequency	Monetary
count	4338.000000	4338.000000	4338.000000
mean	92.536422	91.720609	2054.266460
std	100.014169	228.785094	8989.230441
min	1.000000	1.000000	3.750000
25%	18.000000	17.000000	307.415000
50%	51.000000	41.000000	674.485000
75%	142.000000	100.000000	1661.740000
max	374.000000	7847.000000	280206.020000

```
# Segment Customers based on RFM
rfm['R_Segment'] = pd.qcut(rfm['Recency'], 4, labels=[4, 3, 2, 1])
rfm['F_Segment'] = pd.qcut(rfm['Frequency'], 4, labels=[1, 2, 3, 4])
```

```
rfm['M_Segment'] = pd.qcut(rfm['Monetary'], 4, labels=[1, 2, 3, 4])
rfm['RFM_Score'] = rfm[['R_Segment', 'F_Segment', 'M_Segment']].sum(axis=1)
```

```
rfm.sample(5)
```

CustomerID	Recency	Frequency	Monetary	R_Segment	F_Segment	M_Segment	RFM_Score
14014.0	77	39	505.18	2	2	2	6
17979.0	36	146	757.71	3	4	3	10
17585.0	109	214	1133.07	2	4	3	9
14465.0	247	130	1058.62	1	4	3	8
17505.0	162	52	1145.84	1	3	3	7

```
# Customers with highest RFM Scores
rfm.sort_values('RFM_Score', ascending=False)
```

CustomerID	Recency	Frequency	Monetary	R_Segment	F_Segment	M_Segment	RFM_Score
18198.0	4	159	5425.56	4	4	4	12
18210.0	2	134	2621.38	4	4	4	12
18225.0	3	271	5509.12	4	4	4	12
18283.0	4	756	2094.88	4	4	4	12
16983.0	13	148	1931.25	4	4	4	12
...	...	...	...	...	...	...	...
12402.0	323	11	225.60	1	1	1	3
18185.0	249	17	304.25	1	1	1	3
18190.0	192	15	284.46	1	1	1	3
18191.0	262	7	207.80	1	1	1	3
18193.0	165	16	243.76	1	1	1	3

4338 rows × 7 columns

5. Customer Churn Analysis

```
df4.head(2)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSales	Month
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30	12

```
# Create a basket matrix for association rule mining
customer_last_purchase = df4.groupby("CustomerID")['InvoiceDate'].max()
customer_last_purchase.head(5)
```

CustomerID	InvoiceDate
12346.0	2011-01-18 10:01:00
12347.0	2011-12-07 15:52:00
12348.0	2011-09-25 13:13:00
12349.0	2011-11-21 09:51:00
12350.0	2011-02-02 16:01:00

dtype: datetime64[ns]

```
type(customer_last_purchase)
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None,
fastpath: bool | lib.NoDefault=lib.no_default) -> None

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
```

```
current_date
```

```
Timestamp('2011-12-10 12:50:00')
```

```
customer_last_purchase = (current_date - customer_last_purchase).dt.days
customer_last_purchase.head(5)
```

CustomerID	InvoiceDate
12346.0	326
12347.0	2
12348.0	75
12349.0	19
12350.0	310

dtype: int64

```
# Define churn threshold (e.g., 90 days without purchase)
churn_threshold = 90
churned_customers = customer_last_purchase[customer_last_purchase > churn_threshold]
churned_customers.head(5)
```

CustomerID	InvoiceDate
12346.0	326