💻 Question No: 04

⚙️ Setup

- Ensure the Python kernel has the necessary libraries: `pandas` , `matplotlib` and `lets-plot` , `os` , `numpy` , `statsmodels` , `seaborn`
- Ensure the `Online Retail.xlsx` file is in the `data` folder.

In [34]:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import os
from sklearn import tree
os.getcwd()
import numpy as np
import statsmodels.api as sm

from lets_plot import * # This imports all of ggplot2's functions
LetsPlot.setup_html()
```

Data Cleaning:

Filter the data to exclude negative or zero values for "Quantity" and "UnitPrice". Why is this step necessary?

In [35]:

```
df = pd.read_excel('D:/Data Science for Marketing-I/data/Online Retail.xlsx')
df
```

Out[35]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0
...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0

541909 rows × 8 columns

In [36]:

```python
df.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  object        
 1   StockCode    541909 non-null  object        
 2   Description  540455 non-null  object        
 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  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

In [37]:

```python
df.head()
```

Out[37]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Un Kingc
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Un Kingc
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc

In [38]:

```python
df.describe(include='object')
```

Out[38]:

	InvoiceNo	StockCode	Description	Country
count	541909	541909	540455	541909
unique	25900	4070	4223	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	United Kingdom
freq	1114	2313	2369	495478

In [39]:

```
df = df[df['Quantity']>0]
df = df[df['UnitPrice']>0]
```

💡 the DataFrame to include only rows where the value in the Quantity column is greater than 0,filters the DataFrame to include only rows where the value in the UnitPrice column is greater than 0

In [40]:

```
df
```

Out[40]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0
...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0

530104 rows × 8 columns

Time-Series Analysis of Number of Orders:

Plot the total number of orders over time using a monthly aggregation. Explain how the code achieves this visualization.Why is excluding dates after December 1, 2011 helpful, and how does this impact the analysis?

In [41]:

```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['Month'] = df['InvoiceDate'].dt.to_period('M')

# Monthly number of orders
orders_by_month = df.groupby('Month')['InvoiceNo'].nunique()
```

In [42]:

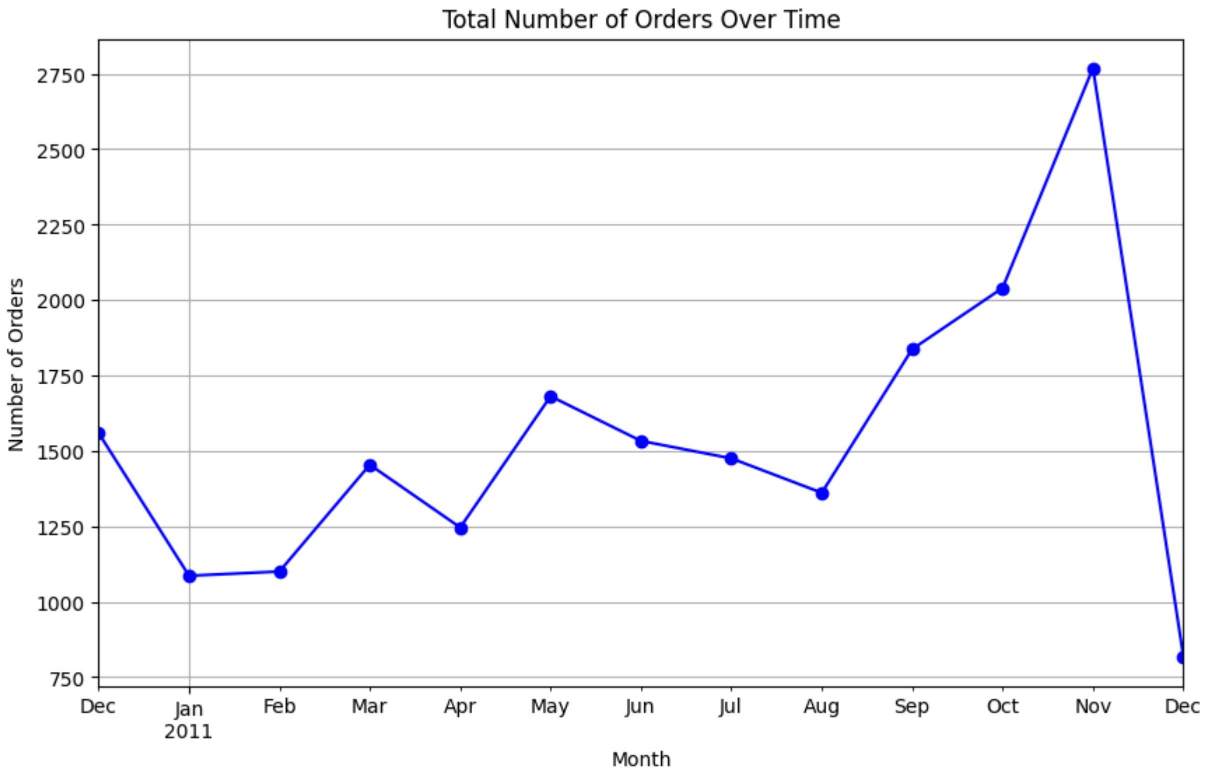
```
orders_by_month
```

Out[42]:

```
Month
2010-12    1559
2011-01    1086
2011-02    1100
2011-03    1454
2011-04    1246
2011-05    1681
2011-06    1533
2011-07    1475
2011-08    1361
2011-09    1837
2011-10    2040
2011-11    2769
2011-12     819
Freq: M, Name: InvoiceNo, dtype: int64
```

In [43]:

```
plt.figure(figsize=(10, 6))
orders_by_month.plot(kind='line', marker='o', color='blue')
plt.title("Total Number of Orders Over Time")
plt.xlabel("Month")
plt.ylabel("Number of Orders")
plt.grid(True)
plt.show()
```



💡 Orders fluctuated at lower levels in the first half of the year, followed by a strong growth phase starting in September. The peak in November suggests a significant seasonal influence. The sharp drop in December might be due to post-peak normalization or seasonality ending.

Revenue Analysis Over Time:

Create a time-series plot for total revenue over time. Describe how revenue is calculated and interpret the trends observed.Why is ax.set_ylim used, and how does adjusting this parameter affect the plot?

In [44]:

```
df=df.loc[df['UnitPrice']>0]
```

In [45]:

```
df['sales']=df['Quantity']*df['UnitPrice']
monthly_revenue_df=df.set_index('InvoiceDate')['sales'].resample('M').sum()
monthly_revenue_df
```

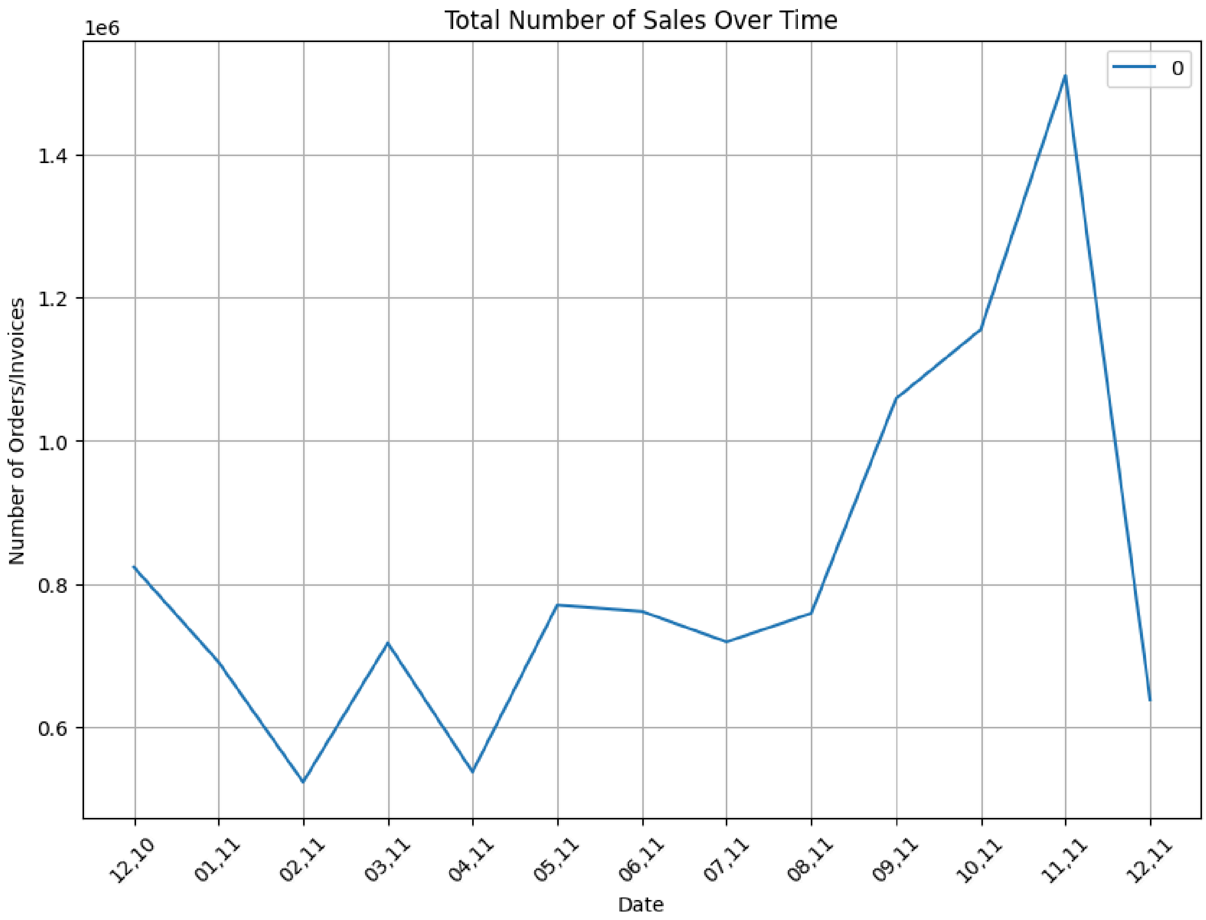
```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\3994170172.py:2: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  monthly_revenue_df=df.set_index('InvoiceDate')['sales'].resample('M').sum()
```

Out[45]:

```
InvoiceDate
2010-12-31     823746.140
2011-01-31     691364.560
2011-02-28     523631.890
2011-03-31     717639.360
2011-04-30     537808.621
2011-05-31     770536.020
2011-06-30     761739.900
2011-07-31     719221.191
2011-08-31     759138.380
2011-09-30    1058590.172
2011-10-31    1154979.300
2011-11-30    1509496.330
2011-12-31     638792.680
Freq: ME, Name: sales, dtype: float64
```

In [46]:

```
# Assuming 'monthly_revenue_df' is already defined and indexed by date
ax = pd.DataFrame(monthly_revenue_df.values).plot(
    grid=True,
    figsize=(10, 7),
)

ax.set_xlabel('Date')
ax.set_ylabel('Number of Orders/Invoices')
ax.set_title('Total Number of Sales Over Time')

# Formatting x-axis ticks for dates
plt.xticks(
    range(len(monthly_revenue_df.index)),
    [x.strftime('%m,%y') for x in monthly_revenue_df.index],
    rotation=45
)

plt.show()
```



💡 An overall upward or downward trend could indicate growth or decline in sales. There is a drop in sales from December 2010 to January 2011

Sales remain relatively stable with small peaks and dips from February to August 2011.During this time, sales fluctuate between 600,000 and 800,000.

Significant Growth:Sales start to rise sharply from September 2011, showing a dramatic increase through October and peaking in November 2011 at approximately 1,500,00

Repeat vs. Unique Customers Analysis:

Analyze and plot the number of repeat and unique customers each month. How are repeat customers identified?Explain the use of a dual-axis plot to show both the customer count and repeat customer percentage. Why is this type of visualization effective?

In [47]:

```
invoice_customerdf1=df.groupby(by=['InvoiceNo','InvoiceDate']).agg({
    'sales':sum,
    'CustomerID':max,
    'Country':max,
}).reset_index()
invoice_customerdf1
```

```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\95178010.py:1: FutureWarning: T
he provided callable <built-in function sum> is currently using SeriesGroupBy.sum. I
n a future version of pandas, the provided callable will be used directly. To keep c
urrent behavior pass the string "sum" instead.
  invoice_customerdf1=df.groupby(by=['InvoiceNo','InvoiceDate']).agg({
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\95178010.py:1: FutureWarning: T
he provided callable <built-in function max> is currently using SeriesGroupBy.max. I
n a future version of pandas, the provided callable will be used directly. To keep c
urrent behavior pass the string "max" instead.
  invoice_customerdf1=df.groupby(by=['InvoiceNo','InvoiceDate']).agg({
```

Out[47]:

	InvoiceNo	InvoiceDate	sales	CustomerID	Country
0	536365	2010-12-01 08:26:00	139.12	17850.0	United Kingdom
1	536366	2010-12-01 08:28:00	22.20	17850.0	United Kingdom
2	536367	2010-12-01 08:34:00	278.73	13047.0	United Kingdom
3	536368	2010-12-01 08:34:00	70.05	13047.0	United Kingdom
4	536369	2010-12-01 08:35:00	17.85	13047.0	United Kingdom
...	...	...	...	...	...
19997	581584	2011-12-09 12:25:00	140.64	13777.0	United Kingdom
19998	581585	2011-12-09 12:31:00	329.05	15804.0	United Kingdom
19999	581586	2011-12-09 12:49:00	339.20	13113.0	United Kingdom
20000	581587	2011-12-09 12:50:00	249.45	12680.0	France
20001	A563185	2011-08-12 14:50:00	11062.06	NaN	United Kingdom

20002 rows × 5 columns

In [48]:

```
df_customer= df[df['CustomerID']==13047.0]['sales'].sum()
df_customer
```

Out[48]:

```
np.float64(3237.54)
```

In [49]:

```
monthly_repeat_customer_df=invoice_customerdf1.set_index('InvoiceDate').groupby([pd
```

```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\3821557082.py:1: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  monthly_repeat_customer_df=invoice_customerdf1.set_index('InvoiceDate').groupby([p
d.Grouper(freq='M'),'CustomerID']).filter(lambda x:len(x)>1) ['CustomerID'].resample
('M').nunique()
```

In [50]:

```
monthly_repeat_customer_df
```

Out[50]:

```
InvoiceDate
2010-12-31    263
2011-01-31    153
2011-02-28    152
2011-03-31    203
2011-04-30    170
2011-05-31    281
2011-06-30    220
2011-07-31    227
2011-08-31    198
2011-09-30    272
2011-10-31    324
2011-11-30    541
2011-12-31    106
Freq: ME, Name: CustomerID, dtype: int64
```

In [51]:

```
monthly_unique_customer_df=df.set_index('InvoiceDate')['CustomerID'].resample('M').
```

```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\672356104.py:1: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  monthly_unique_customer_df=df.set_index('InvoiceDate')['CustomerID'].resample
('M').nunique()
```

Repeat customers percentage

In [52]:

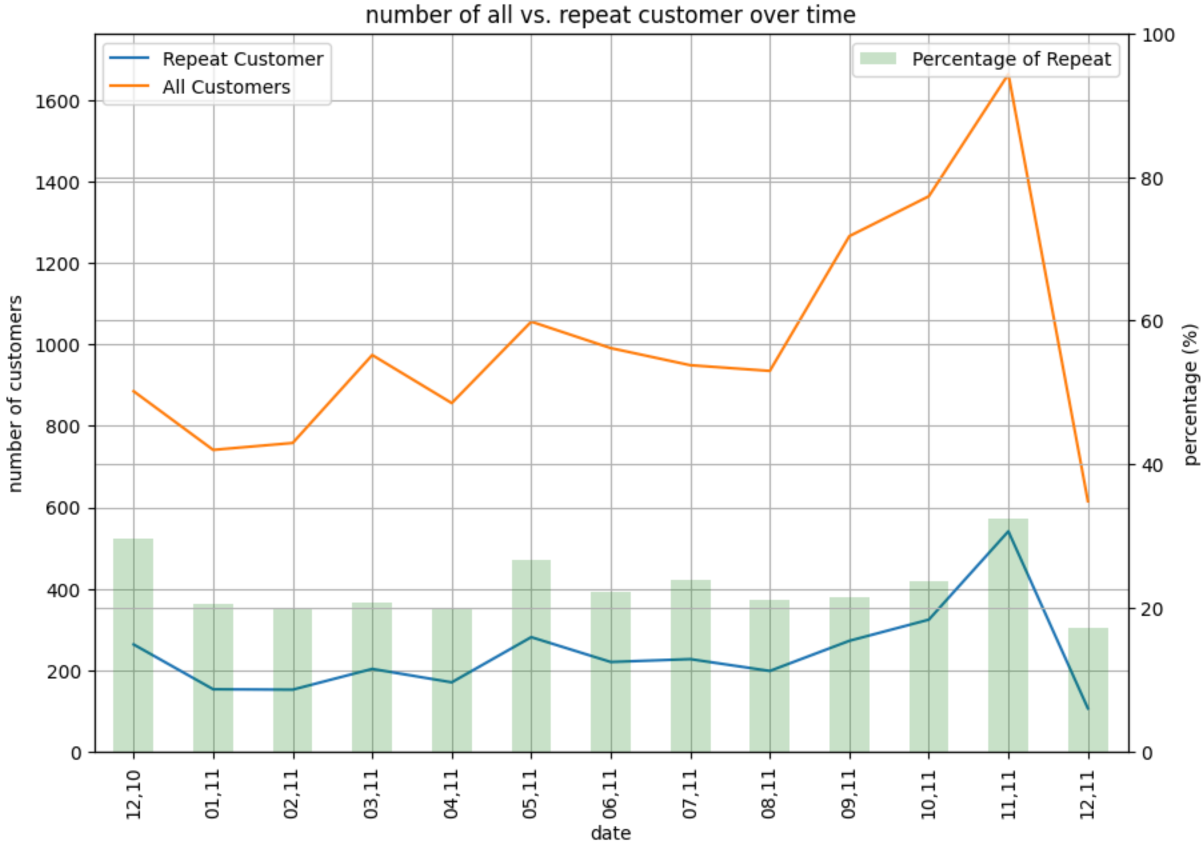
```
monthly_repeat_percentage=monthly_repeat_customer_df/monthly_unique_customer_df*100
monthly_repeat_percentage
```

Out[52]:

```
InvoiceDate
2010-12-31    29.717514
2011-01-31    20.647773
2011-02-28    20.052770
2011-03-31    20.841889
2011-04-30    19.859813
2011-05-31    26.609848
2011-06-30    22.199798
2011-07-31    23.919916
2011-08-31    21.176471
2011-09-30    21.484992
2011-10-31    23.753666
2011-11-30    32.512019
2011-12-31    17.235772
Freq: ME, Name: CustomerID, dtype: float64
```

In [53]:

```
ax = pd.DataFrame(monthly_repeat_customer_df.values).plot(
    figsize=(10,7)
)
pd.DataFrame(monthly_unique_customer_df.values).plot(
    ax=ax,
    grid=True
)

ax2=pd.DataFrame(monthly_repeat_percentage.values).plot.bar(
    ax=ax,
    grid=True,
    secondary_y=True, # graph la right side y axis varum (secoundary_y = true)
    color='green',
    alpha=0.2

)
ax.set_xlabel('date')
ax.set_ylabel('number of customers')
ax.set_title('number of all vs. repeat customer over time')

ax2.set_ylabel('percentage (%)')

ax.legend(['Repeat Customer','All Customers'])
ax2.legend(['Percentage of Repeat'],loc='upper right')

ax.set_ylim([0,monthly_unique_customer_df.values.max()+100])
ax2.set_ylim([0,100])

plt.xticks( #position
    range(len(monthly_repeat_customer_df.index)),
    [x.strftime('%m,%y') for x in monthly_repeat_customer_df.index],# convert the t
    rotation=45
)
plt.show()
```



💡 * Repeat customers are identified by checking if a customer ID appears in previous months, indicating they have made multiple purchases.

- A dual-axis plot shows the total customer count on one axis and the repeat customer percentage on another, allowing easy comparison of both metrics over time.

- This visualization is effective because it provides a clear view of the relationship between the total customer base and the retention rate, helping identify trends in customer loyalty.

Revenue from Repeat Customers:

Compute and plot monthly revenue from repeat customers. Why is it important to track this metric separately from overall revenue?How would this information influence business strategies for customer retention?

In [54]:

```
monthly_rev_repeat_customer_df=invoice_customerdf1.set_index('InvoiceDate').groupby
monthly_rev_repeat_customer_df
```

```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\469262344.py:1: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  monthly_rev_repeat_customer_df=invoice_customerdf1.set_index('InvoiceDate').groupb
y([pd.Grouper(freq='M'),'CustomerID']).filter(lambda x:len(x)>1).resample('M').sum()
['sales']
```

Out[54]:

```
InvoiceDate
2010-12-31    359170.60
2011-01-31    222124.00
2011-02-28    191067.27
2011-03-31    267390.48
2011-04-30    195474.18
2011-05-31    378197.04
2011-06-30    376307.26
2011-07-31    317475.00
2011-08-31    317134.25
2011-09-30    500663.36
2011-10-31    574006.87
2011-11-30    713775.85
2011-12-31    146833.97
Freq: ME, Name: sales, dtype: float64
```

In [55]:

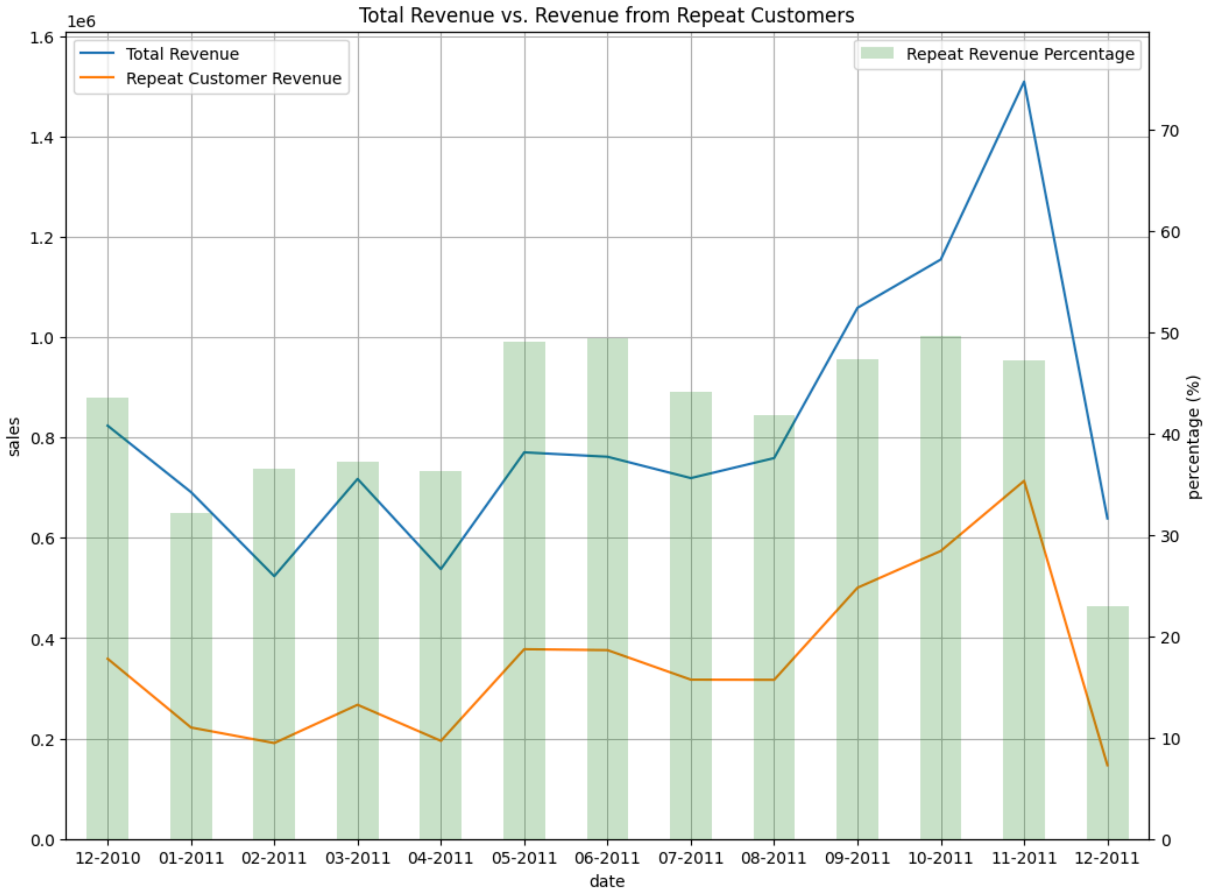
```
monthly_rev_perc_repeat_customer_df=monthly_rev_repeat_customer_df/monthly_revenue_
monthly_rev_perc_repeat_customer_df
```

Out[55]:

```
InvoiceDate
2010-12-31    43.602098
2011-01-31    32.128346
2011-02-28    36.488853
2011-03-31    37.259729
2011-04-30    36.346420
2011-05-31    49.082331
2011-06-30    49.401017
2011-07-31    44.141497
2011-08-31    41.775552
2011-09-30    47.295296
2011-10-31    49.698455
2011-11-30    47.285696
2011-12-31    22.986170
Freq: ME, Name: sales, dtype: float64
```

In [56]:

```python
ax = pd.DataFrame(monthly_revenue_df.values).plot(figsize=(12,9))

pd.DataFrame(monthly_rev_repeat_customer_df.values).plot(
    ax=ax,
    grid=True,
)

ax.set_xlabel('date')
ax.set_ylabel('sales')
ax.set_title('Total Revenue vs. Revenue from Repeat Customers')

ax.legend(['Total Revenue', 'Repeat Customer Revenue'])

ax.set_ylim([0, max(monthly_revenue_df.values)+100000])

ax2 = ax.twinx()
pd.DataFrame(monthly_rev_perc_repeat_customer_df.values).plot(
    ax=ax2,
    kind='bar',
    color='g',
    alpha=0.2
)

ax2.set_ylim([0, max(monthly_rev_perc_repeat_customer_df.values)+30])
ax2.set_ylabel('percentage (%)')
ax2.legend(['Repeat Revenue Percentage'])

ax2.set_xticklabels([
    x.strftime('%m-%Y') for x in monthly_rev_perc_repeat_customer_df.index
])


plt.show()
```



💡 The repeat revenue percentage values (8.0 and 6.5) suggest that a notable portion of total revenue comes from repeat customers.

The graph likely shows how repeat customer revenue contributes to overall revenue over time, helping to identify trends and the effectiveness of customer retention strategies.

Popular Items Over Time:

Analyze monthly sales data for different products and create a plot that visualizes the quantity trends over time

In [57]:

```
date_item_df = pd.DataFrame(
    df.set_index('InvoiceDate').groupby([
        pd.Grouper(freq='M'), 'StockCode'
    ])['Quantity'].sum()
)

date_item_df
```

```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\814040035.py:3: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  pd.Grouper(freq='M'), 'StockCode'
```

Out[57]:

		Quantity
InvoiceDate	**StockCode**	
2010-12-31	**10002**	251
	10120	16
	10125	154
	10133	130
	10135	411
...	**...**	...
2011-12-31	**DCGSSBOY**	1
	DOT	16
	M	819
	POST	124
	gift_0001_10	1

34069 rows × 1 columns

In [58]:

```
# Rank items by the last month sales
last_month_sorted_df = date_item_df.loc['2011-11-30'].sort_values(
    by='Quantity', ascending=False
).reset_index()

last_month_sorted_df
```

Out[58]:

	StockCode	Quantity
0	23084	14954
1	22197	12460
2	22086	7908
3	85099B	5909
4	22578	5366
...	...	...
2934	35638A	1
2935	35638B	1
2936	35818B	1
2937	90152A	1
2938	90160B	1

2939 rows × 2 columns

highlighting the top 5 items with a custom date format and labeled axes

In [59]:

```
date_item_df = pd.DataFrame(
df.loc[
df['StockCode'].isin([23084, 84836, 22197, 23086, '8500981'])
]. set_index('InvoiceDate').groupby([
pd.Grouper(freq='M'),'StockCode'
])['Quantity'].sum()
)
date_item_df
```

```
C:\Users\Mr. Royal\AppData\Local\Temp\ipykernel_6612\3086855463.py:5: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  pd.Grouper(freq='M'),'StockCode'
```

Out[59]:

		Quantity
InvoiceDate	**StockCode**	
2010-12-31	**22197**	2738
	84836	530
2011-01-31	**22197**	1824
	84836	318
2011-02-28	**22197**	2666
	84836	440
2011-03-31	**22197**	2803
	84836	292
2011-04-30	**22197**	1869
	84836	330
2011-05-31	**22197**	6849
	23084	1131
	23086	112
	84836	412
2011-06-30	**22197**	2095
	23084	1713
	23086	37
	84836	352
2011-07-31	**22197**	1876
	23084	294
	23086	129
	84836	446
2011-08-31	**22197**	5421
	23084	1847
	23086	338
	84836	535
2011-09-30	**22197**	4196
	23084	215
	23086	294

		Quantity
InvoiceDate	StockCode	
	84836	527
2011-10-31	22197	5907
	23084	6323
	23086	19
	84836	911
2011-11-30	22197	12460
	23084	14954
	23086	264
	84836	782
2011-12-31	22197	6217
	23084	4311
	23086	35
	84836	69

In [60]:

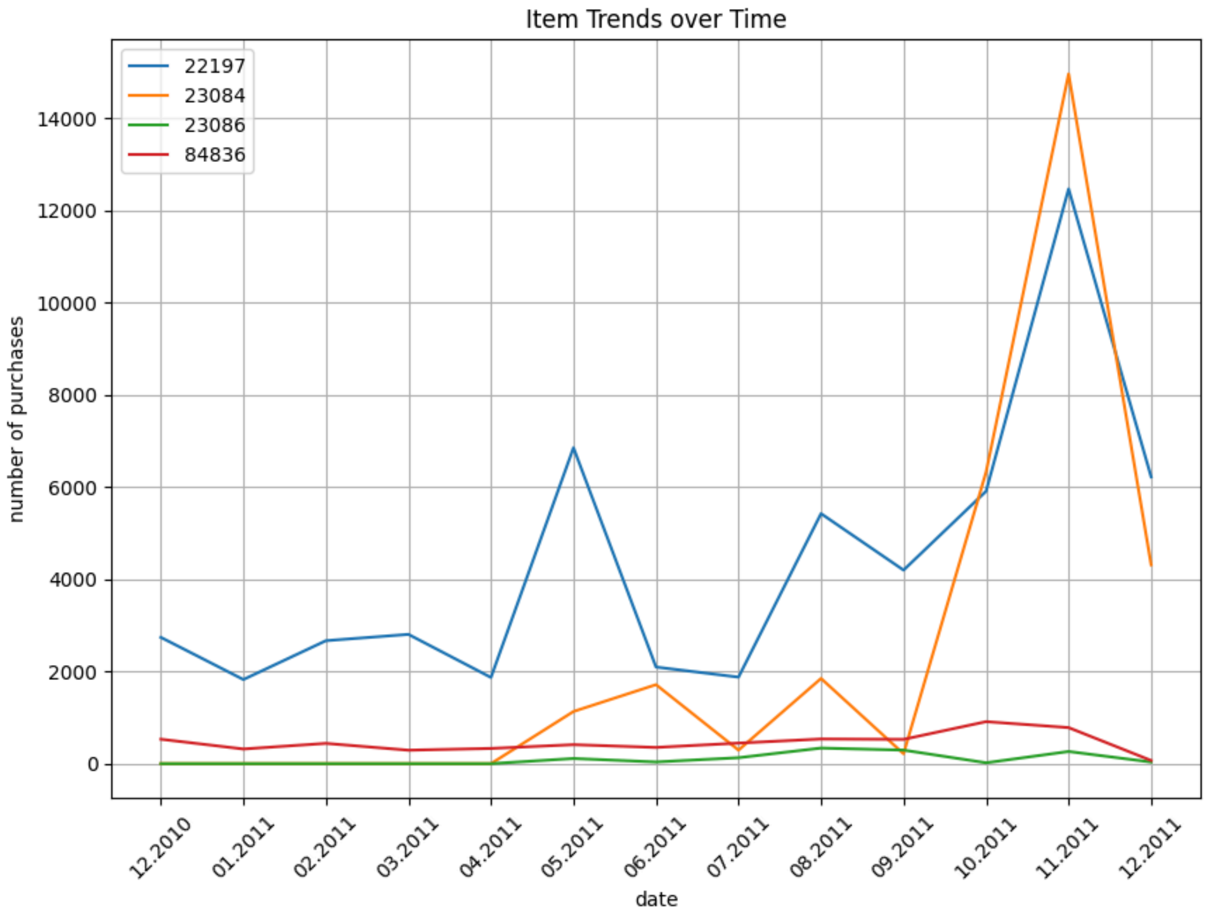
```python
trending_items_df = date_item_df.reset_index().pivot(
    index='InvoiceDate',
    columns='StockCode',
    values='Quantity'
).fillna(0)

# Resetting and setting the index
trending_items_df = trending_items_df.reset_index()
trending_items_df = trending_items_df.set_index('InvoiceDate')

# Check if the columns are a MultiIndex before attempting to drop levels
if isinstance(trending_items_df.columns, pd.MultiIndex):
    trending_items_df.columns = trending_items_df.columns.droplevel(0)

trending_items_df
```

Out[60]:

StockCode	22197	23084	23086	84836
InvoiceDate				
2010-12-31	2738.0	0.0	0.0	530.0
2011-01-31	1824.0	0.0	0.0	318.0
2011-02-28	2666.0	0.0	0.0	440.0
2011-03-31	2803.0	0.0	0.0	292.0
2011-04-30	1869.0	0.0	0.0	330.0
2011-05-31	6849.0	1131.0	112.0	412.0
2011-06-30	2095.0	1713.0	37.0	352.0
2011-07-31	1876.0	294.0	129.0	446.0
2011-08-31	5421.0	1847.0	338.0	535.0
2011-09-30	4196.0	215.0	294.0	527.0
2011-10-31	5907.0	6323.0	19.0	911.0
2011-11-30	12460.0	14954.0	264.0	782.0
2011-12-31	6217.0	4311.0	35.0	69.0

In [61]:

```
ax = pd.DataFrame(trending_items_df.values).plot(
    figsize=(10, 7),
    grid=True,
)

ax.set_ylabel('number of purchases')
ax.set_xlabel('date')
ax.set_title('Item Trends over Time')

ax.legend(trending_items_df.columns, loc='upper left')

plt.xticks(
    range(len(trending_items_df.index)),
    [x.strftime('%m.%Y') for x in trending_items_df.index],
    rotation=45
)

plt.show()
```



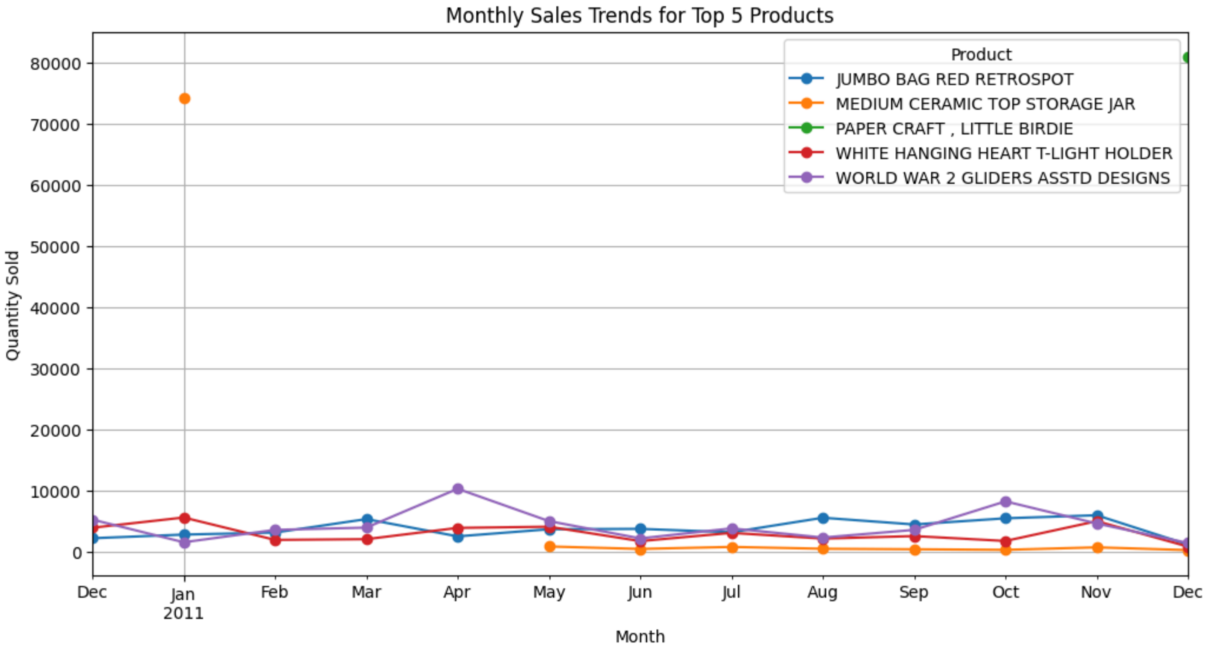
💡 The graph provides insights into purchasing trends over time, helping to identify seasonal patterns, growth or decline in sales, and item performance.

It can highlight anomalies or unusual events that may require further investigation.

Businesses can use this data to optimize inventory management, marketing strategies, and product offerings to align with customer demand.

In [62]:

```
# Aggregate product sales by month
product_sales = df.groupby(['Month', 'Description'])['Quantity'].sum().reset_index(

# Identify top 5 products overall
top_products = product_sales.groupby('Description')['Quantity'].sum().nlargest(5).i

# Filter for top products
top_product_sales = product_sales[product_sales['Description'].isin(top_products)]

# Pivot for visualization
pivot = top_product_sales.pivot(index='Month', columns='Description', values='Quant

# Plot trends for top 5 products
pivot.plot(kind='line', figsize=(12, 6), marker='o')
plt.title("Monthly Sales Trends for Top 5 Products")
plt.ylabel("Quantity Sold")
plt.xlabel("Month")
plt.grid()
plt.legend(title="Product")
plt.show()
```



💡

Outlier in January 2011:

MEDIUM CERAMIC TOP STORAGE JAR (orange line) has a significant spike in sales, with over 80,000 units sold. This appears to be an anomaly, as it drastically exceeds all other monthly sales for any product. WORLD WAR 2 GLIDERS ASSTD DESIGNS (purple line) shows periodic spikes, particularly in April, September, and November. JUMBO BAG RED RETROSPOT (blue line) maintains steady sales throughout the year, with a slight peak in November.

Seasonal Trends:

November emerges as a strong sales month across multiple products, likely due to holiday demand or promotions. Sales for all products sharply decline in December 2011, mirroring seasonal trends observed in other analyses.