Revou Mini Course Case Project

Context

Exploring company sales data reveals important insights into market trends and product performance. This analysis is crucial for making strategic decisions and identifying opportunities for growth and improvement.

Questions to be Answered

> 1. Which product lines have the highest and the lowest sales? Create the chart that is represetable
> 2. Show sales performance over time? Is there any pattern?
> 3. How does deal size correlate with total sales? What is the percentage of the contribution for each type of deal?

In [29]:
```
# Import all libraries needed and load the data

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sales = pd.read_csv('/Users/raffaelnathanielsiregar/Downloads/sales_data.csv')
sales.head()
```

Out[29]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERDATE	STATUS	PRODUCTLINE	PRODUCTCODE	CUSTOMERNAME	CITY	DEALSIZE
0	10100	30	100.00	1/6/2003 0:00	Shipped	Vintage Cars	S18_1749	Online Diecast Creations Co.	Nashua	Medium
1	10100	50	67.80	1/6/2003 0:00	Shipped	Vintage Cars	S18_2248	Online Diecast Creations Co.	Nashua	Medium
2	10100	22	86.51	1/6/2003 0:00	Shipped	Vintage Cars	S18_4409	Online Diecast Creations Co.	Nashua	Small
3	10100	49	34.47	1/6/2003 0:00	Shipped	Vintage Cars	S24_3969	Online Diecast Creations Co.	Nashua	Small
4	10101	25	100.00	1/9/2003 0:00	Shipped	Vintage Cars	S18_2325	Blauer See Auto, Co.	Frankfurt	Medium

In [30]:
```
# inspect the dataframe in order to check null-values and column data type

sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2824 entries, 0 to 2823
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ORDERNUMBER      2824 non-null   int64  
 1   QUANTITYORDERED  2824 non-null   int64  
 2   PRICEEACH        2824 non-null   float64
 3   ORDERDATE        2824 non-null   object 
 4   STATUS           2824 non-null   object 
 5   PRODUCTLINE      2824 non-null   object 
 6   PRODUCTCODE      2824 non-null   object 
 7   CUSTOMERNAME     2824 non-null   object 
 8   CITY             2824 non-null   object 
 9   DEALSIZE         2824 non-null   object 
dtypes: float64(1), int64(2), object(7)
memory usage: 220.8+ KB
```

In [31]:
```
# adjust columns name for more readable name and use case flexibility

sales.columns = ['order_number', 'quantity_ordered', 'price_each', 'order_date', 'status', 'product_line', 'product_code', 'customer_nam
sales.head()
```

Out[31]:

	order_number	quantity_ordered	price_each	order_date	status	product_line	product_code	customer_name	city	deal_size
0	10100	30	100.00	1/6/2003 0:00	Shipped	Vintage Cars	S18_1749	Online Diecast Creations Co.	Nashua	Medium
1	10100	50	67.80	1/6/2003 0:00	Shipped	Vintage Cars	S18_2248	Online Diecast Creations Co.	Nashua	Medium
2	10100	22	86.51	1/6/2003 0:00	Shipped	Vintage Cars	S18_4409	Online Diecast Creations Co.	Nashua	Small
3	10100	49	34.47	1/6/2003 0:00	Shipped	Vintage Cars	S24_3969	Online Diecast Creations Co.	Nashua	Small
4	10101	25	100.00	1/9/2003 0:00	Shipped	Vintage Cars	S18_2325	Blauer See Auto, Co.	Frankfurt	Medium

In [33]:
```
# Deal with data types

for col in sales.columns:
  if col == 'order_number' or col == 'quantity_ordered' or col == 'price_each':
    continue
  elif col == 'order_date':
    sales[col] = pd.to_datetime(sales[col])
  else:
    sales[col] = sales[col].astype('category')

sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2824 entries, 0 to 2823
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   order_number      2824 non-null   int64
 1   quantity_ordered  2824 non-null   int64
 2   price_each        2824 non-null   float64
 3   order_date        2824 non-null   datetime64[ns]
 4   status            2824 non-null   category
 5   product_line      2824 non-null   category
 6   product_code      2824 non-null   category
 7   customer_name     2824 non-null   category
 8   city              2824 non-null   category
 9   deal_size         2824 non-null   category
dtypes: category(6), datetime64[ns](1), float64(1), int64(2)
memory usage: 115.9 KB
```

In [34]:

```python
# Inspect duplicated values

sales.duplicated().value_counts()
```

Out[34]:

```
False    2823
True        1
Name: count, dtype: int64
```

In [35]:

```python
# Deal with duplicated values

sales = sales.drop_duplicates()
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2823 entries, 0 to 2823
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   order_number      2823 non-null   int64
 1   quantity_ordered  2823 non-null   int64
 2   price_each        2823 non-null   float64
 3   order_date        2823 non-null   datetime64[ns]
 4   status            2823 non-null   category
 5   product_line      2823 non-null   category
 6   product_code      2823 non-null   category
 7   customer_name     2823 non-null   category
 8   city              2823 non-null   category
 9   deal_size         2823 non-null   category
dtypes: category(6), datetime64[ns](1), float64(1), int64(2)
memory usage: 137.8 KB
```

1. Which product lines have the highest and the lowest sales? Create the chart that is represetable

In [36]:

```python
# Multiply quantity_ordered column by price_each to get total sales each transaction

sales['revenue'] = sales.quantity_ordered * sales.price_each
sales.head()
```

Out[36]:

	order_number	quantity_ordered	price_each	order_date	status	product_line	product_code	customer_name	city	deal_size	revenue
0	10100	30	100.00	2003-01-06	Shipped	Vintage Cars	S18_1749	Online Diecast Creations Co.	Nashua	Medium	3000.00
1	10100	50	67.80	2003-01-06	Shipped	Vintage Cars	S18_2248	Online Diecast Creations Co.	Nashua	Medium	3390.00
2	10100	22	86.51	2003-01-06	Shipped	Vintage Cars	S18_4409	Online Diecast Creations Co.	Nashua	Small	1903.22
3	10100	49	34.47	2003-01-06	Shipped	Vintage Cars	S24_3969	Online Diecast Creations Co.	Nashua	Small	1689.03
4	10101	25	100.00	2003-01-09	Shipped	Vintage Cars	S18_2325	Blauer See Auto, Co.	Frankfurt	Medium	2500.00

> In order to know which product lines have the highest and the lowest sales, a column called 'revenue' is needed. The 'revenue' column is obtained by multiplying 'price_each' column and 'quantity_ordered' column.

In [37]:

```python
# Ensuring the data type of revenue column

sales.revenue.dtypes
```

Out[37]:

```
dtype('float64')
```

In [38]:

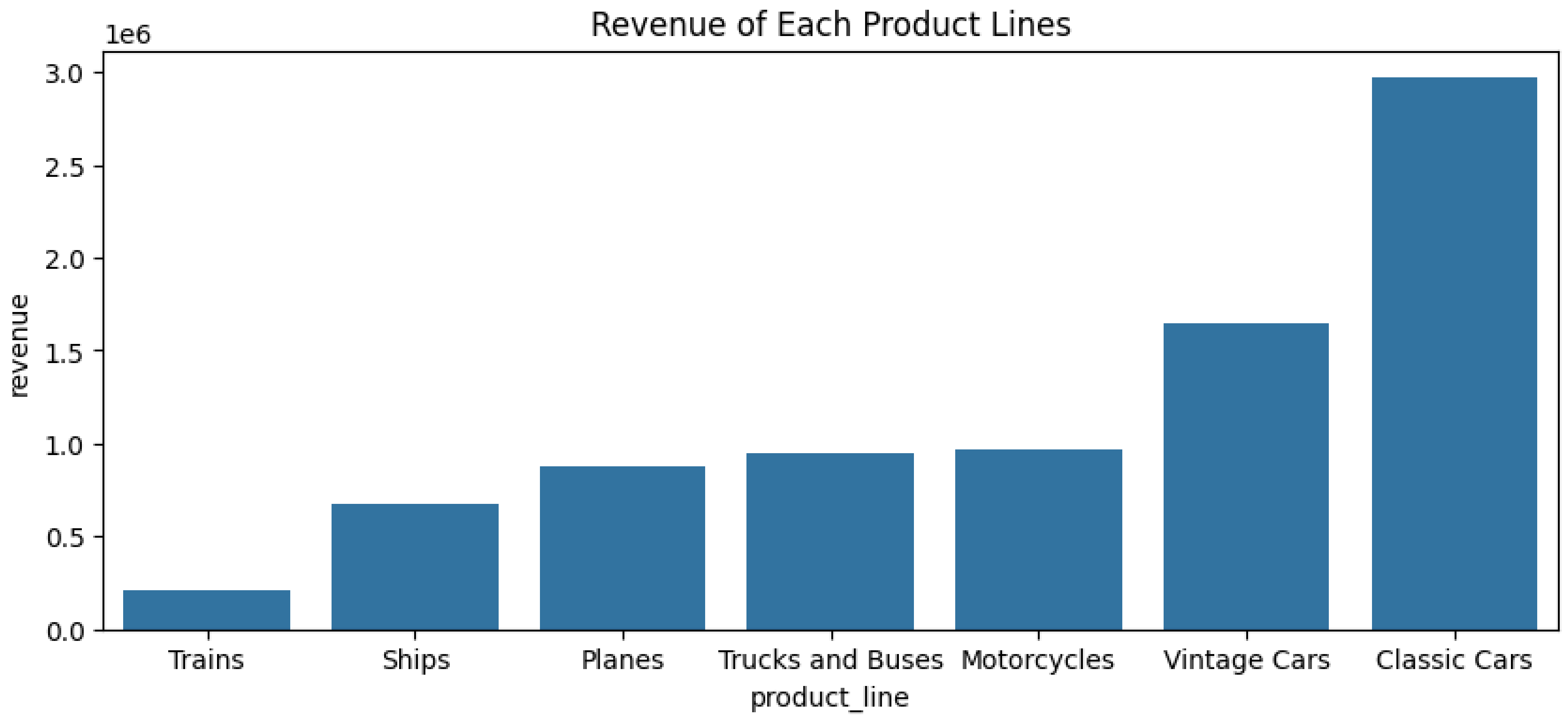
```python
# Grouping the revenue by product_line

agg_prod_line = sales.groupby('product_line', observed=False).agg({'revenue': 'sum'}).reset_index()
agg_prod_line = agg_prod_line.sort_values(by = 'revenue', ascending=False, ignore_index=True)
agg_prod_line
```

Out[38]:

	product_line	revenue
0	Classic Cars	2968546.40
1	Vintage Cars	1644212.05
2	Motorcycles	971086.29
3	Trucks and Buses	947355.18
4	Planes	877942.21
5	Ships	677940.40
6	Trains	203804.26

In [52]:

```
# Visualize each product_line 's revenue comparison

plt.figure(figsize=(10,4))
sns.barplot(agg_prod_line, x = 'product_line', y='revenue', order = agg_prod_line.sort_values('revenue').product_line)
plt.title('Revenue of Each Product Lines')
plt.show()
```



> From the bar plot above, it is shown that "Trains" has the lowest revenue and "Classic Cars" has the highest revenue.

2. Show sales performance over time? Is there any pattern?

In [41]:

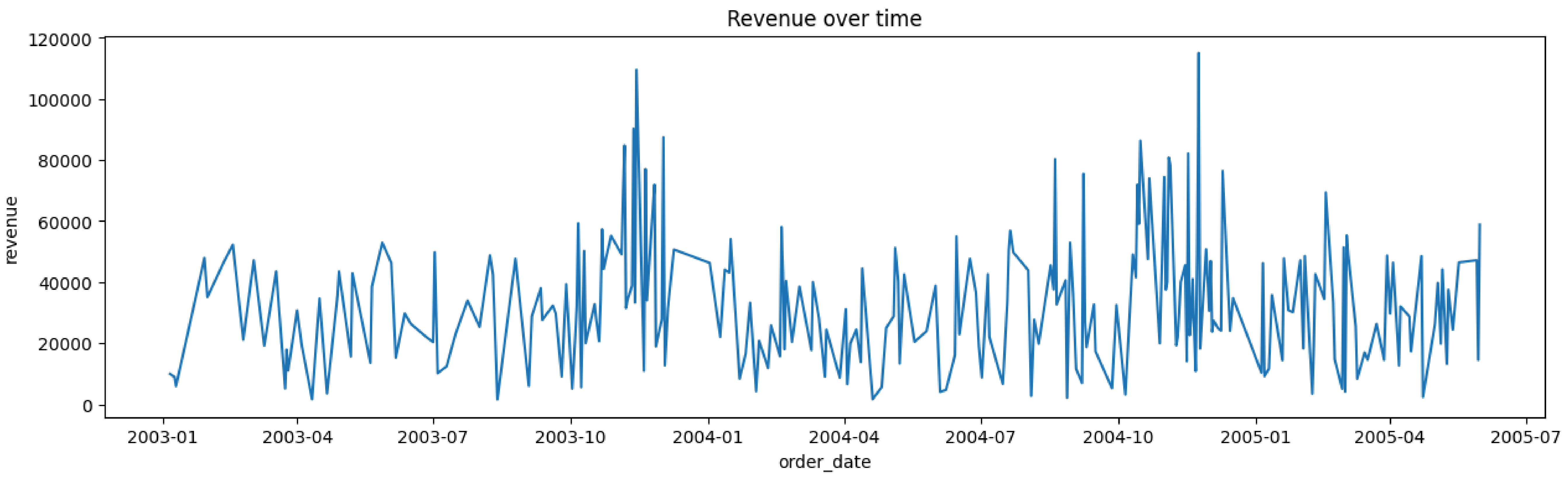
```
# Grouping the total revenue by date

agg_date_day = sales.groupby('order_date').agg({'revenue': 'sum'}).reset_index().sort_values(by='order_date', ascending=True)
agg_date_day.head()
```

Out[41]:

	order_date	revenue
0	2003-01-06	9982.25
1	2003-01-09	8976.96
2	2003-01-10	5955.74
3	2003-01-29	47886.21
4	2003-01-31	35084.80

In [42]:

```
# Visualize total revenue over time

plt.figure(figsize=(15,4))
sns.lineplot(agg_date_day, x='order_date', y='revenue')
plt.title('Revenue over time')
plt.show()
```



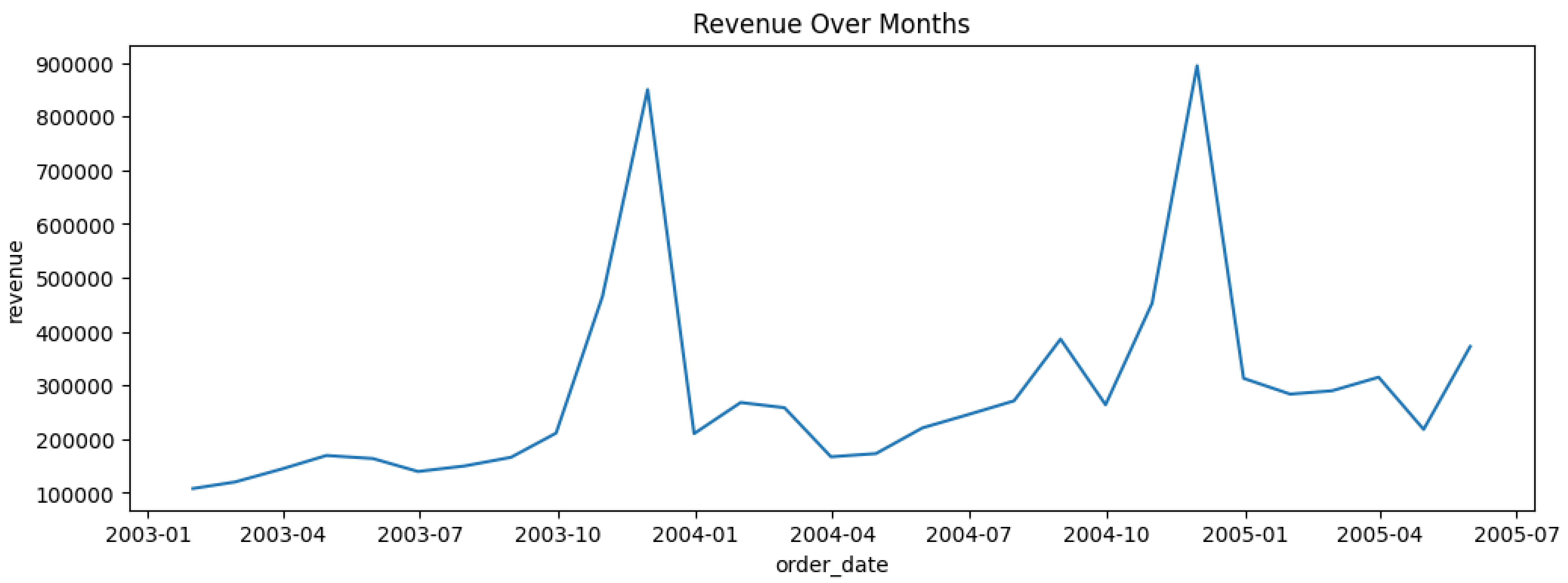
> According to the chart above, there are a significant increase towards the end of each year

In [43]:
```python
# Grouping the total revenue by each month in order to get the total revenue trend

sales['order_date'] = pd.to_datetime(sales['order_date'])
agg_date_month = sales.groupby(pd.Grouper(key='order_date', freq='ME')).agg({'revenue': 'sum'}).reset_index().sort_values(by='order_date
agg_date_month.head(12)
```

Out[43]:

	order_date	revenue
0	2003-01-31	107885.96
1	2003-02-28	120036.80
2	2003-03-31	144096.23
3	2003-04-30	169421.03
4	2003-05-31	163654.12
5	2003-06-30	139552.84
6	2003-07-31	149869.73
7	2003-08-31	166026.32
8	2003-09-30	211045.86
9	2003-10-31	466240.57
10	2003-11-30	850203.27
11	2003-12-31	210117.21

In [44]:
```python
# Visualize revenue over months

plt.figure(figsize=(12,4))
sns.lineplot(agg_date_month, x = 'order_date', y='revenue')
plt.title('Revenue Over Months')
plt.show()
```



> Creating a total revenue chart with monthly timeframe can give another point of view. It's clearer that there are a significant increase during the end of each year. It also shows that every year, the total revenue are keep increasing

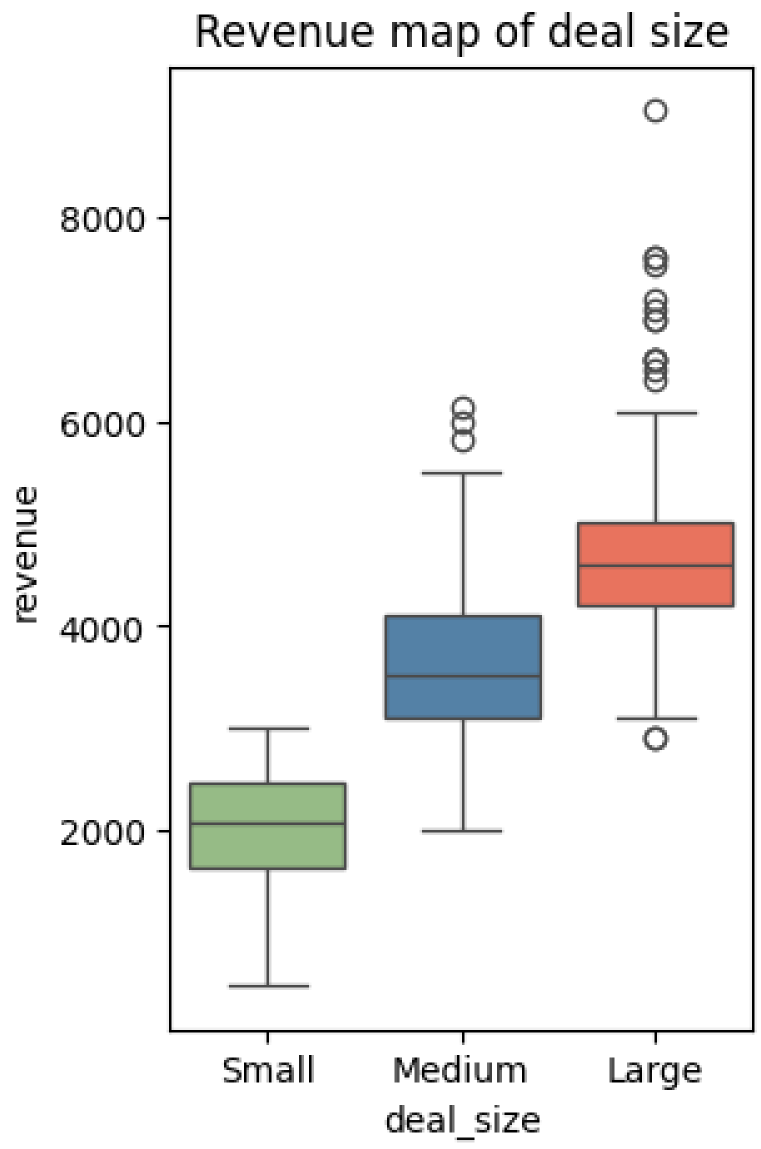
3. How does deal size correlate with total sales? What is the percentage of the contribution for each type of deal?

In [45]:
```python
# Encode the deal size into numeric values

sales['deal_size_encoded'] = sales['deal_size'].map({'Small': 1, 'Medium': 2, 'Large': 3})
dealsize_revenue_corr = sales[['deal_size_encoded', 'revenue']].corr()
dealsize_revenue_corr
```

Out[45]:

	deal_size_encoded	revenue
deal_size_encoded	1.000000	0.785638
revenue	0.785638	1.000000

In [46]:
```python
# Visualize correlation between 3 different deal size with the revenue

plt.figure(figsize=(3, 5))
custom_palette = ['#FF6347', '#4682B4', '#93C47D']
plt.title('Revenue map of deal size')
sns.boxplot(data=sales,
            x='deal_size',
            y='revenue',
            order=sales.sort_values('deal_size_encoded', ascending=False).deal_size,
            palette = custom_palette,
            hue='deal_size')
plt.show()
```



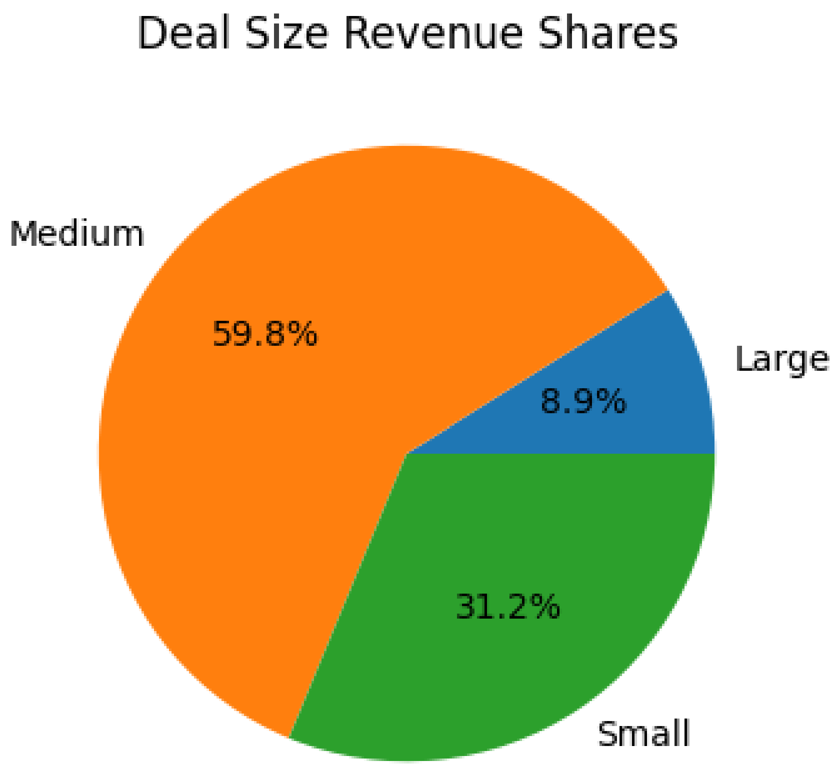
> The Boxplot above gives us the information of the correlation between deal size and revenue. "Small" is categorized with average revenue about 2000 and "Medium" deal size is categorized with average revenue about 3500. Meanwhile, "Large" deal size is categorized with average revenue about 4500.

In [47]:
```
# Create revenue shares for each product line

agg_dealsize_revenue = sales.groupby('deal_size', observed=False).agg({'revenue': 'sum'}).reset_index()
agg_dealsize_revenue
```

Out[47]:

	deal_size	revenue
0	Large	738757.91
1	Medium	4961736.68
2	Small	2590392.20

In [48]:
```
# Visualize Deal Size Revenue shares

plt.figure(figsize=(4,4))
plt.title('Deal Size Revenue Shares')
plt.pie(agg_dealsize_revenue['revenue'], labels = agg_dealsize_revenue['deal_size'], autopct='%1.1f%%')
plt.show()
```



> From the figure above we can conclude that even though "Large" deal size has higher average of revenue, "Large" deal size can not beat the revenue shares of "Small and "Medium" deal size. "Medium" deal_size is leading the revenue shares with 59.8%. Followed by "Small" deal size and "Large" deal size for 31.2% and 8.9% each.

Conclusions

> 1. "Classic Cars" has the highest revenue. In contrast, "Trains" has the lowest revenue
> 2. The revenue increase significantly by the end of each year. The revenue trend is still growing up over time
> 3. Deal size represent the size of revenue for each transaction. "Medium" deal size has the highest revenue shares.