Data Analysis

- Name: **Ankit Verma**
- Mobile no: **+91-7004518207**
- Email: **ankitv2524@gmail.com**

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [17]:
```
sales = pd.read_csv("/Users/ankit/Job_assignments_new/data/sales_data_sample.csv")
sales.head()
```

Out[17]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	ST/
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shi
1	10121	34	81.35	5	2765.90	5/7/2003 00:00	Shi
2	10134	41	94.74	2	3884.34	7/1/2003 00:00	Shi
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shi
4	10159	49	100.00	14	5205.27	10/10/2003 00:00	Shi

5 rows × 25 columns

In [180…
```
# to check the ytpe of sales
type(sales)
```

Out[180…
```
pandas.core.frame.DataFrame
```

In [183…
```
for c in sales.columns:
    print(f'Number of {c} unique values: {sales[c].nunique()}')
```

```
Number of ORDERNUMBER unique values: 307
Number of QUANTITYORDERED unique values: 58
Number of PRICEEACH unique values: 1016
Number of ORDERLINENUMBER unique values: 18
Number of SALES unique values: 2763
Number of ORDERDATE unique values: 252
Number of STATUS unique values: 6
```

```
Number of QTR_ID unique values: 4
Number of MONTH_ID unique values: 12
Number of YEAR_ID unique values: 3
Number of PRODUCTLINE unique values: 7
Number of MSRP unique values: 80
Number of PRODUCTCODE unique values: 109
Number of CUSTOMERNAME unique values: 92
Number of PHONE unique values: 91
Number of ADDRESSLINE1 unique values: 92
Number of ADDRESSLINE2 unique values: 9
Number of CITY unique values: 73
Number of STATE unique values: 16
Number of POSTALCODE unique values: 73
Number of COUNTRY unique values: 19
Number of TERRITORY unique values: 3
Number of CONTACTLASTNAME unique values: 77
Number of CONTACTFIRSTNAME unique values: 72
Number of DEALSIZE unique values: 3
Number of YEAR_MONTH unique values: 29
```

In [184…

```
sales.describe()
```

Out[184…

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	Q
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.0(
mean	10258.725115	35.092809	83.658544	6.466171	3553.889072	2.7
std	92.085478	9.741443	20.174277	4.225841	1841.865106	1.2(
min	10100.000000	6.000000	26.880000	1.000000	482.130000	1.0(
25%	10180.000000	27.000000	68.860000	3.000000	2203.430000	2.0(
50%	10262.000000	35.000000	95.700000	6.000000	3184.800000	3.0(
75%	10333.500000	43.000000	100.000000	9.000000	4508.000000	4.0(
max	10425.000000	97.000000	100.000000	18.000000	14082.800000	4.0(

In [24]:

```
# This is used to print information about dataframe including dtype ,column and non-
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
```

```
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

Descriptive stats include those that summarise central tendancy,dispersion and shape of dataset excluding NaN values.

In [26]:
```
sales.describe()
```

Out[26]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	Q
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.0(
mean	10258.725115	35.092809	83.658544	6.466171	3553.889072	2.7
std	92.085478	9.741443	20.174277	4.225841	1841.865106	1.2(
min	10100.000000	6.000000	26.880000	1.000000	482.130000	1.0(
25%	10180.000000	27.000000	68.860000	3.000000	2203.430000	2.0(
50%	10262.000000	35.000000	95.700000	6.000000	3184.800000	3.0(
75%	10333.500000	43.000000	100.000000	9.000000	4508.000000	4.0(
max	10425.000000	97.000000	100.000000	18.000000	14082.800000	4.0(

To check the no columns

It is used to print the array of columns present in the dataset.

In [28]:
```
sales.columns
```

Out[28]:
```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
       'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
       'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
       'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
       'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
       'DEALSIZE'],
      dtype='object')
```

In [31]:
```
# to check the shape
sales.shape
```

Out[31]:
```
(2823, 25)
```

In [36]:
```
# print the array of countries present in this dataset under column COUNTRY.
sales["COUNTRY"].unique()
```

Out[36]:
```
array(['USA', 'France', 'Norway', 'Australia', 'Finland', 'Austria', 'UK',
       'Spain', 'Sweden', 'Singapore', 'Canada', 'Japan', 'Italy',
       'Denmark', 'Belgium', 'Philippines', 'Germany', 'Switzerland',
       'Ireland'], dtype=object)
```

Exploratory Analysis and Visualization

SALES:

Sales is one of the important factor to look at for which we will visualise scatterplot with help of year and status column.

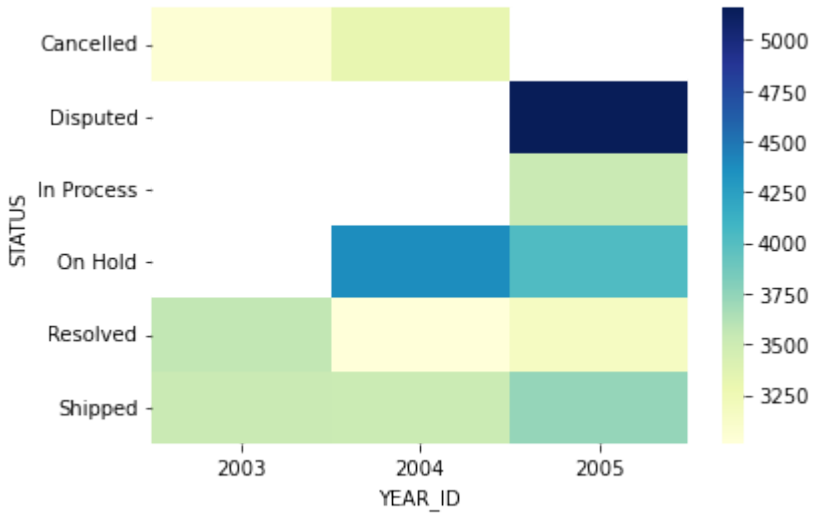
In [38]:
```
df1=sales[['STATUS','YEAR_ID','SALES']]
df1.head()
```

Out[38]:

	STATUS	YEAR_ID	SALES
0	Shipped	2003	2871.00
1	Shipped	2003	2765.90
2	Shipped	2003	3884.34
3	Shipped	2003	3746.70
4	Shipped	2003	5205.27

In [40]:
```
heatmap_data=pd.pivot_table(df1,values='SALES',index=['STATUS'],columns='YEAR_ID')
sns.heatmap(heatmap_data,cmap='YlGnBu')
```

Out[40]:
```
<AxesSubplot:xlabel='YEAR_ID', ylabel='STATUS'>
```



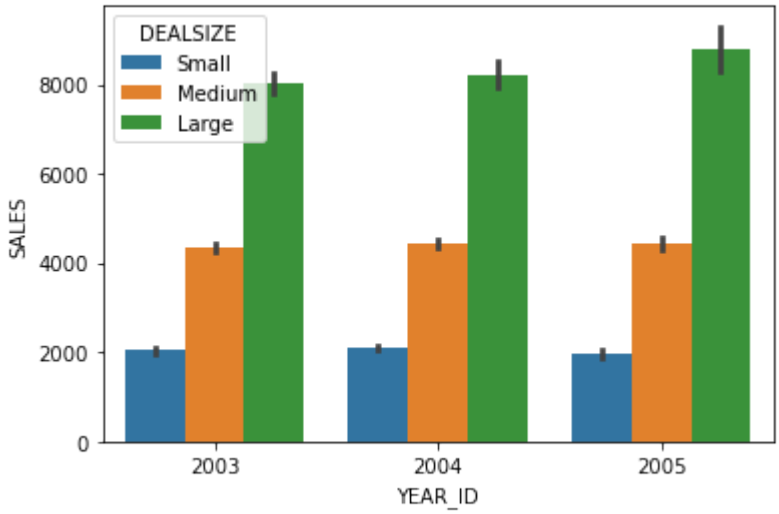
Based on the scatterplot, it is evident that brighter colors correspond to higher sales for a given order status. Consequently, the scatterplot indicates that sales peaked in 2005 when the status was 'Disputed'.

DEALSIZE

In this analysis, we will use a barplot to determine which DEALSIZE category achieved the highest sales each year.

In [43]:
```
sns.barplot('YEAR_ID','SALES',hue='DEALSIZE',data=sales)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



The graph shows that the 'large' DEALSIZE consistently has the highest sales each year, with its peak sales occurring in 2005.

ORDER PER YEAR

In this analysis, we will use a pie chart to identify the year with the highest number of orders.

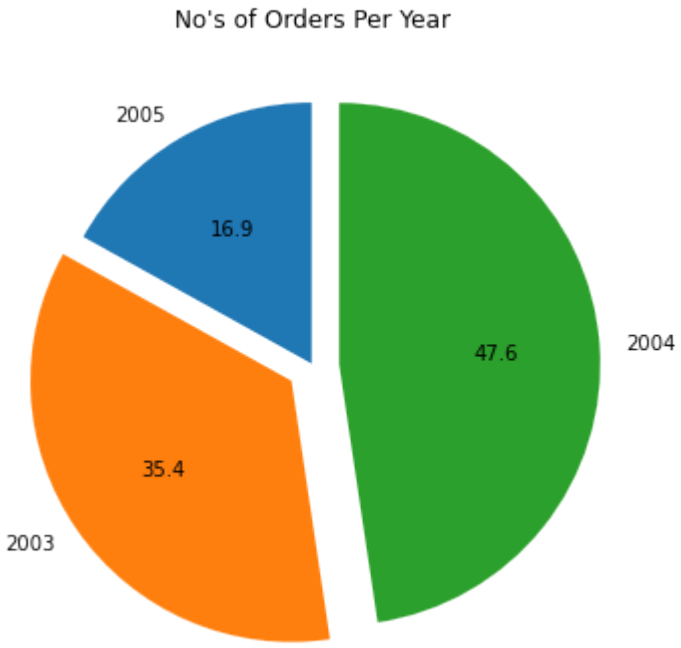
In [46]:
```
sales.YEAR_ID.nunique()
```

Out[46]:
```
3
```

In [54]:
```
sales_analysis = sales.YEAR_ID.value_counts().sort_values(ascending = True)
sales_analysis
```

Out[54]:
```
2005     478
2003    1000
2004    1345
Name: YEAR_ID, dtype: int64
```

In [68]:
```
plt.figure(figsize=(12,6))
plt.title("No's of Orders Per Year")
plt.pie(sales_analysis,labels=sales_analysis.index,autopct='%1.1f', explode=(0.01,0.
```



The data indicates that the highest number of orders was in 2004.

country

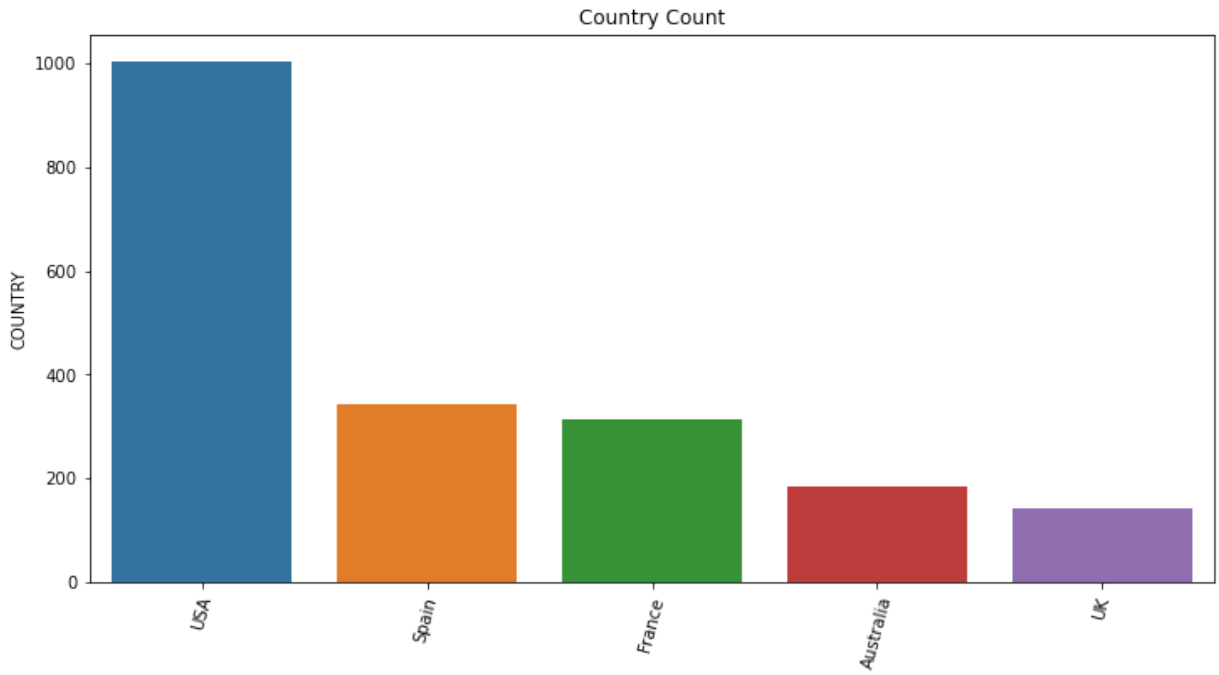
In this analysis, we will identify the top 5 countries with the highest number of orders.

In [71]:
```
top_5_countries = sales.COUNTRY.value_counts().head(5)
top_5_countries
```

Out[71]:
```
USA          1004
Spain         342
France        314
Australia     185
UK            144
Name: COUNTRY, dtype: int64
```

In [81]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=75)
plt.title("Country Count")
sns.barplot(top_5_countries.index, top_5_countries);
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



The data clearly shows that the maximum number of orders came from the **USA** , followed by **Spain** and top 5th is **uk** .

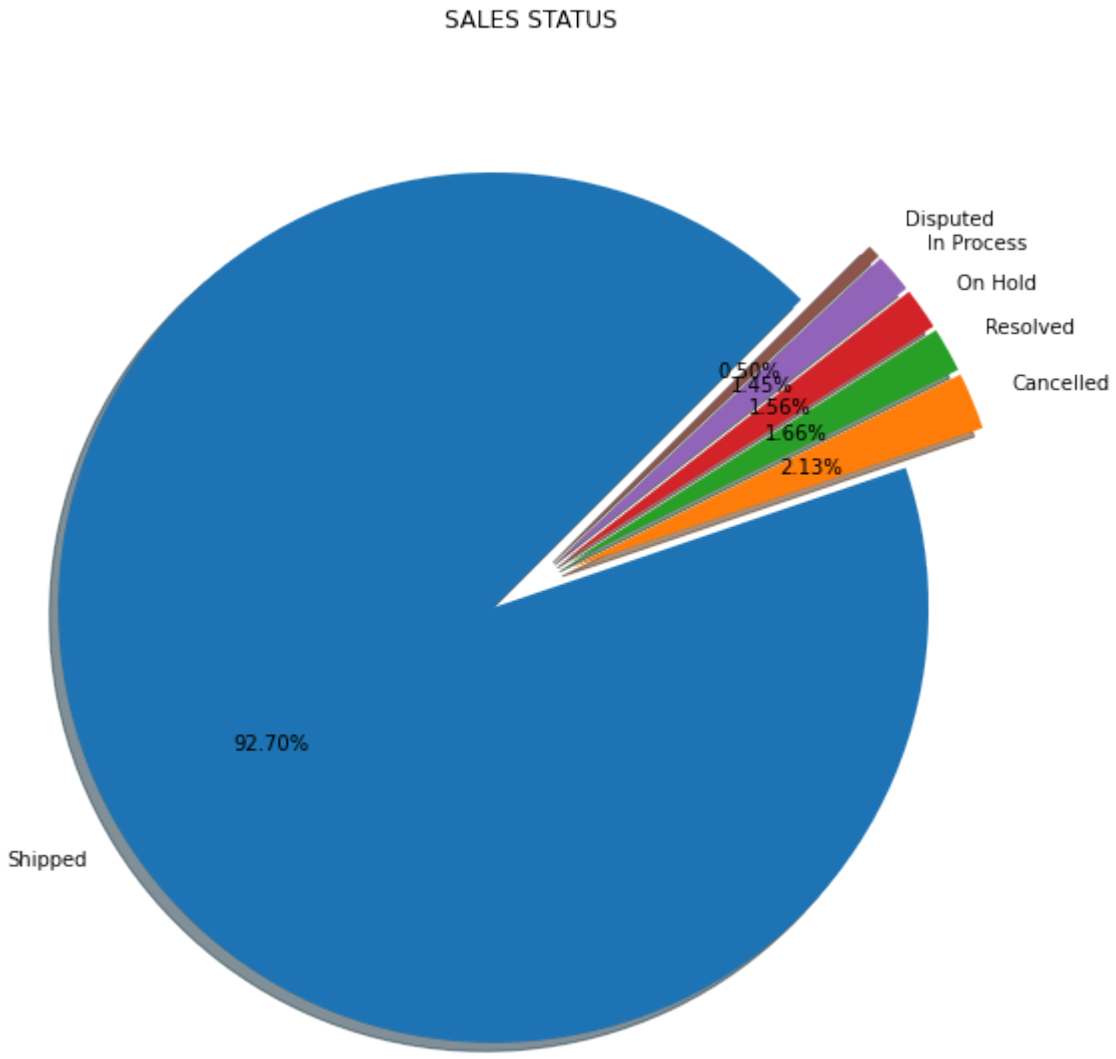
STATUS

In this analysis, we will visualize the status of orders using a pie chart.

In [86]:
```
status = sales.STATUS.value_counts().sort_values(ascending = False)
status
```

Out[86]:
```
Shipped       2617
Cancelled       60
Resolved        47
On Hold         44
In Process      41
Disputed        14
Name: STATUS, dtype: int64
```

In [96]:
```
plt.figure(figsize=(10,10))
plt.title('SALES STATUS')
plt.pie(status, labels=status.index, autopct='%1.2f%%',shadow=True,explode=(0.1,0.1,
```



According to the pie chart, it's evident that the most common order status is 'Shipped' with approx of 92 % share, while 'Disputed' accounts for the least number of orders with 0.5 % share.

> **highest proportion when top 15 orders arrange according to QUANTITYORDERED ?**

Here we will first use .head to take top 15 data.

In [101…

```
top_15 = sales.sort_values('QUANTITYORDERED',ascending=False,  ignore_index = True).
top_15[['DEALSIZE', 'QUANTITYORDERED']]
```

Out[101…

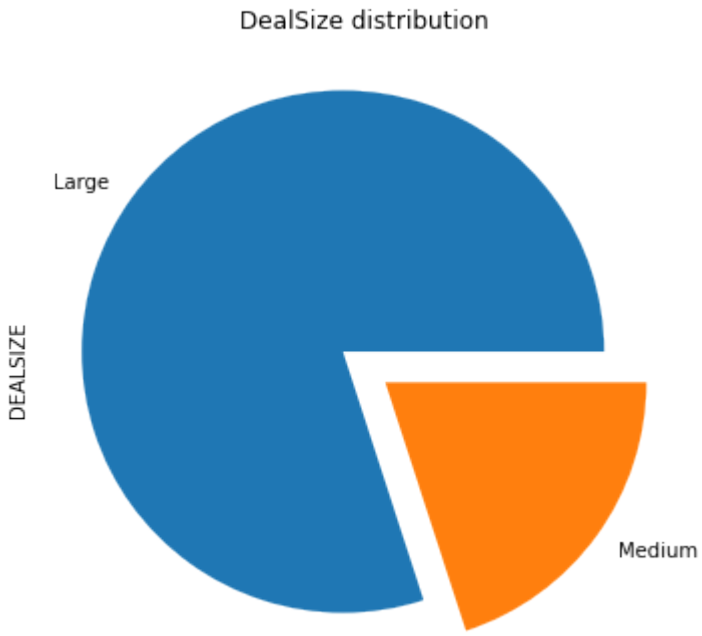
	DEALSIZE	QUANTITYORDERED
0	Large	97
1	Large	85
2	Large	77
3	Large	76
4	Large	76
5	Large	76
6	Large	70
7	Large	70

	DEALSIZE	QUANTITYORDERED
8	Large	66
9	Large	66
10	Medium	66
11	Large	66
12	Medium	66
13	Large	65
14	Medium	64

here, we will use graph to check the proportion of DEALSIZE.

In [111…

```
plt.figure(figsize=(9,6))
top_15['DEALSIZE'].value_counts(normalize = True).plot(kind = 'pie', explode=(0.1,0.
plt.title('DealSize distribution')
plt.show()
```



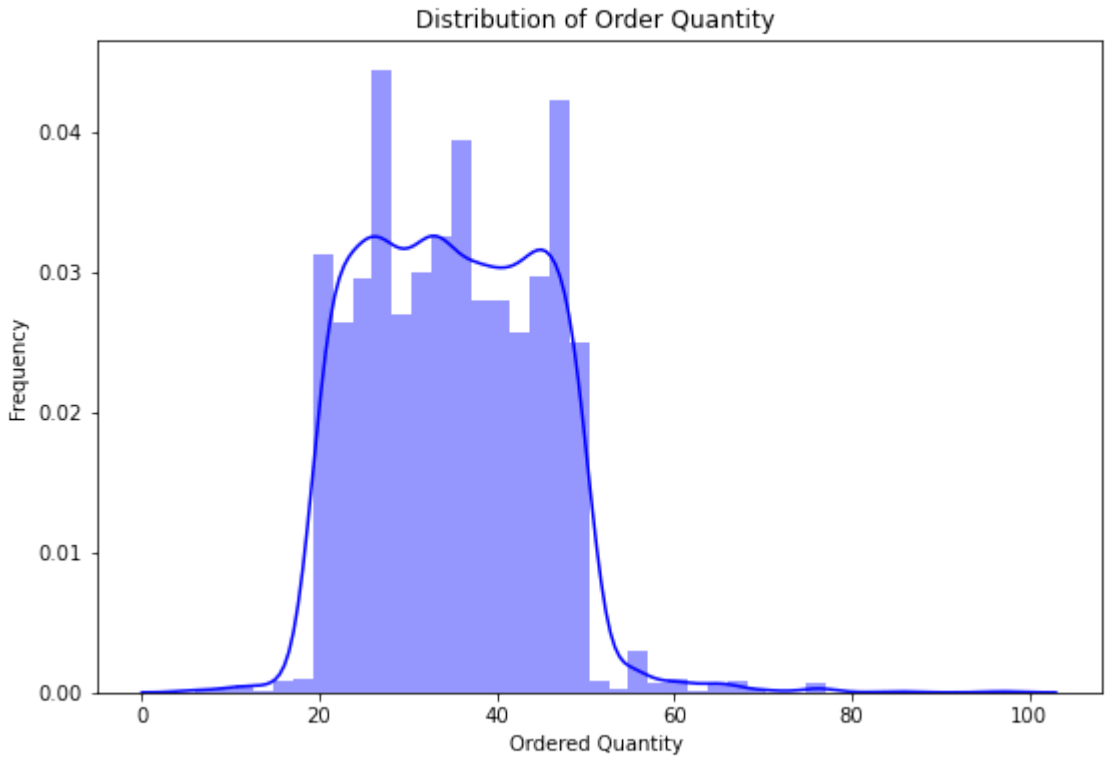
From the graph we get to know that Medium DEALSIZE has highest proportion.

> **From the graph show the frequency of QUANTITYORDERED and tell when it is high**

In [117…

```
plt.figure(figsize=(9,6))
sns.distplot(sales['QUANTITYORDERED'],color='b')
plt.title('Distribution of Order Quantity ')
plt.xlabel('Ordered Quantity ')
plt.ylabel('Frequency')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



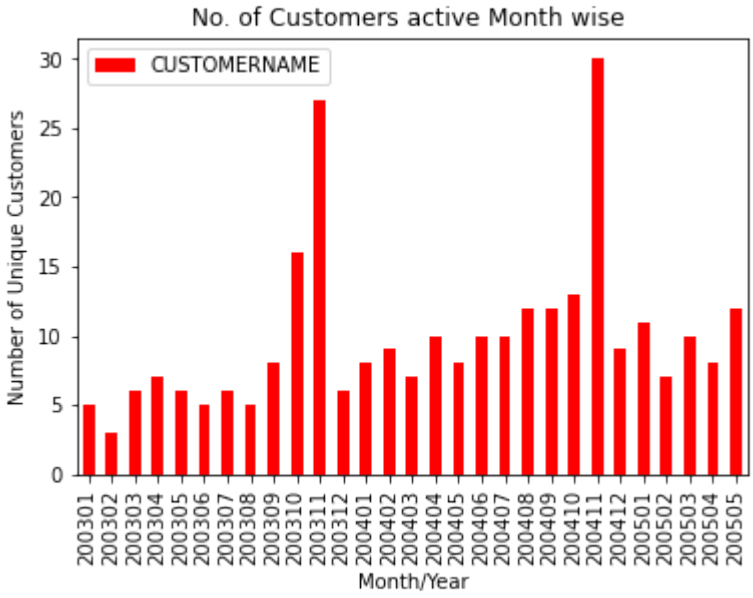
The data clearly indicates that the frequency of QUANTITYORDERED is highest when its value is 20, and there's also a notable peak in frequency within the range of 40 to 60.

In which month customers are highly active

In [126…

```
plt.figure(figsize=(25,25))
sales['YEAR_MONTH'] = sales['YEAR_ID'].map(str)+sales['MONTH_ID'].map(str).map(lambd
monthly_active = sales.groupby(['YEAR_MONTH'])['CUSTOMERNAME'].nunique().reset_index
monthly_active.plot(kind='bar',x='YEAR_MONTH',y='CUSTOMERNAME', color = "r")
plt.title('No. of Customers active Month wise ')
plt.xlabel('Month/Year')
plt.ylabel('Number of Unique Customers')
plt.xticks(rotation=90)
plt.show()
```

```
<Figure size 1800x1800 with 0 Axes>
```



Show where the monthly revenue growth rate is high.

In [129…

```
revenue_months_wise = sales.groupby(['YEAR_ID','MONTH_ID'])['SALES'].sum().reset_ind
revenue_months_wise
```

Out[129…

	YEAR_ID	MONTH_ID	SALES
0	2003	1	129753.60
1	2003	2	140836.19
2	2003	3	174504.90
3	2003	4	201609.55
4	2003	5	192673.11
5	2003	6	168082.56
6	2003	7	187731.88
7	2003	8	197809.30
8	2003	9	263973.36
9	2003	10	568290.97
10	2003	11	1029837.66
11	2003	12	261876.46
12	2004	1	316577.42
13	2004	2	311419.53
14	2004	3	205733.73
15	2004	4	206148.12
16	2004	5	273438.39
17	2004	6	286674.22
18	2004	7	327144.09
19	2004	8	461501.27
20	2004	9	320750.91
21	2004	10	552924.25
22	2004	11	1089048.01
23	2004	12	372802.66
24	2005	1	339543.42
25	2005	2	358186.18
26	2005	3	374262.76
27	2005	4	261633.29
28	2005	5	457861.06

In [131…

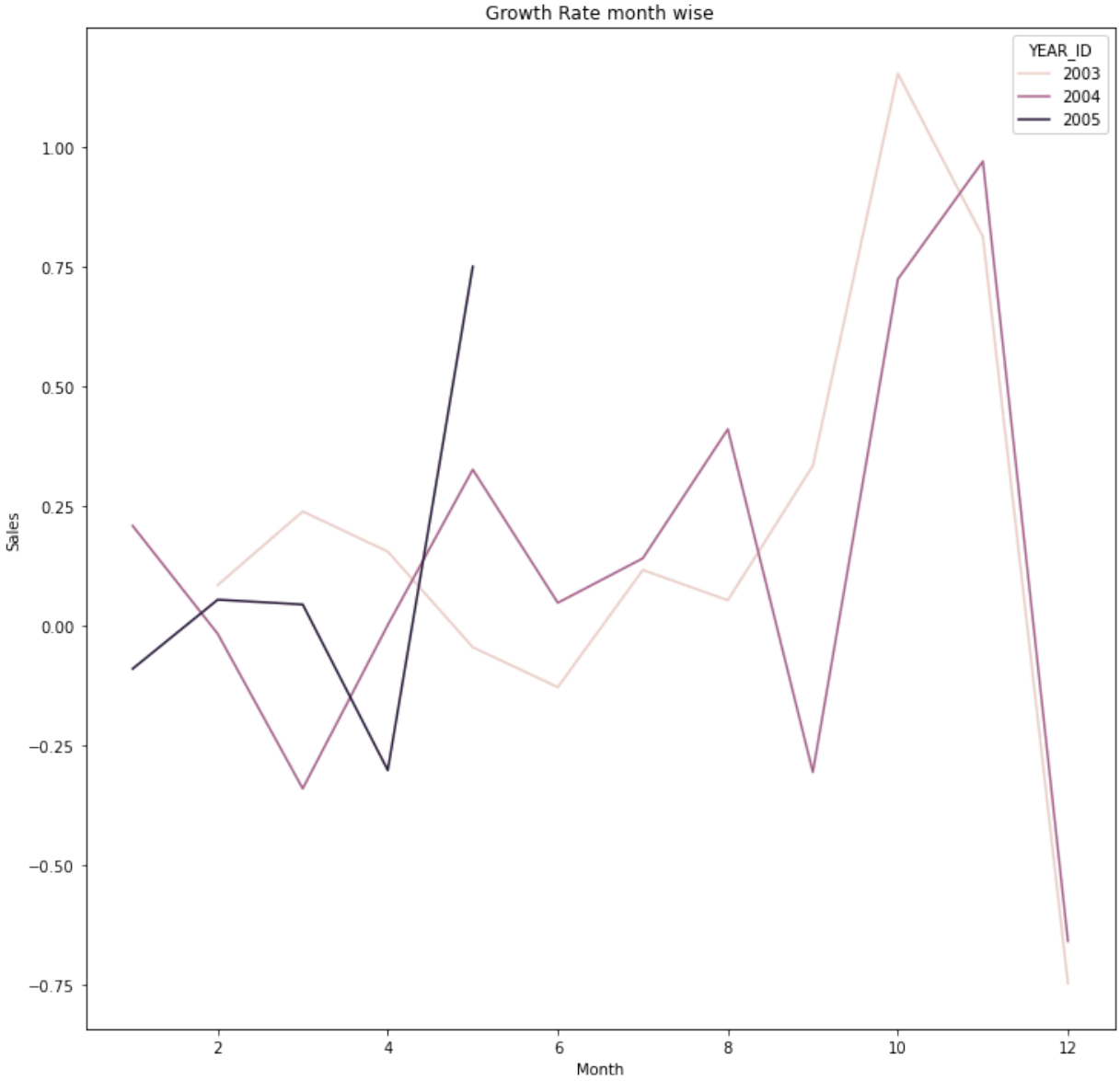
```
revenue_months_wise['MONTHLY GROWTH'] = revenue_months_wise['SALES'].pct_change()
revenue_months_wise.head()
```

Out[131…

	YEAR_ID	MONTH_ID	SALES	MONTHLY GROWTH
0	2003	1	129753.60	NaN
1	2003	2	140836.19	0.085413
2	2003	3	174504.90	0.239063
3	2003	4	201609.55	0.155323
4	2003	5	192673.11	-0.044325

In [135…

```
plt.figure(figsize=(12,12))
sns.lineplot(x="MONTH_ID", y="MONTHLY GROWTH",hue="YEAR_ID", data=revenue_months_wis
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Growth Rate month wise')
plt.show()
```



> **To identify the countries with the highest sales, we'll consider only those countries with more than 50 responses.**

In [171…

```
# First we will use 'groupby' data frame method to aggregate the rows for each count
countries = sales.groupby('COUNTRY')[['SALES']].mean().sort_values('SALES',ascending
# countries
```

Now we will filter the result to get countries of only those - which have more than 50 responses.

In [185…
```
top_sale = countries.loc[sales.COUNTRY.value_counts()>50]
top_sale
```

Out[185…

COUNTRY	SALES
Denmark	3899.002381
Sweden	3684.459825
Austria	3673.864182
Singapore	3651.752025
Japan	3618.611731
Norway	3617.220000
USA	3613.528715
Finland	3582.412065
Germany	3556.001452
Spain	3554.640117
France	3537.950701
Australia	3408.773514
UK	3325.558750
Italy	3315.701858
Canada	3201.122286

So country having maximum sales is Denmark.

#Gain the insights¶

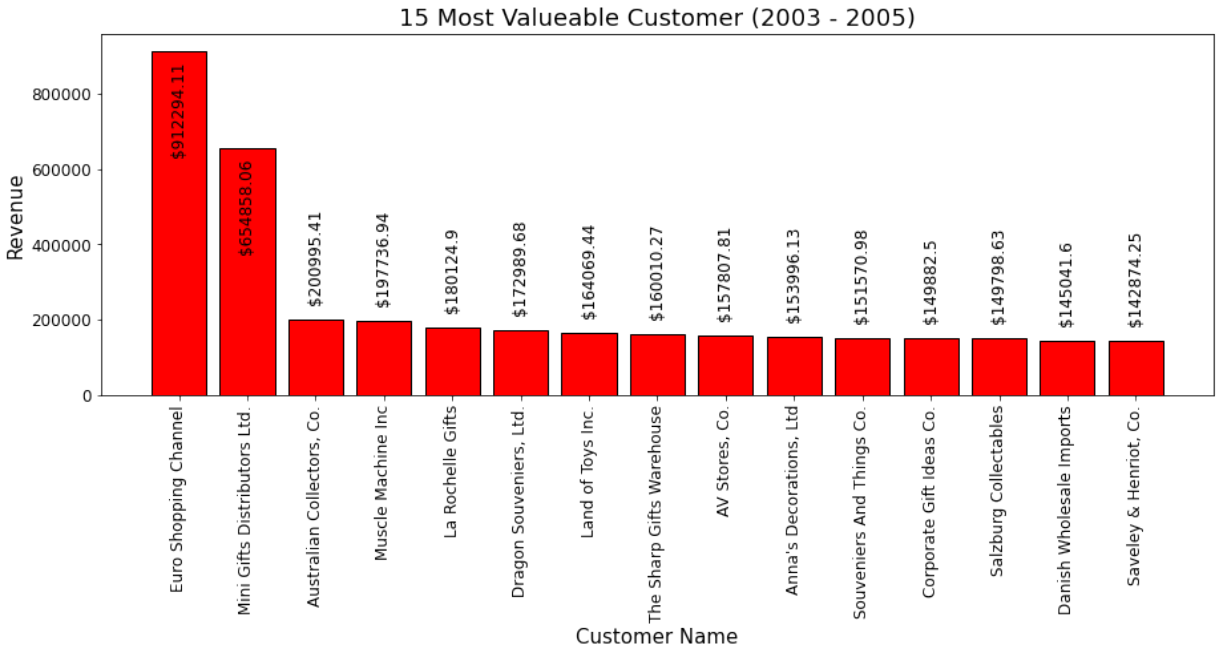
Find out 15 Most Valuable Customers

The Most Valuable Customers are the customer who are the most profitable for a company (have a big sales on them). These customers buy more or higher-value than the other customers.

In [206…
```
top_customer = sales.groupby(['CUSTOMERNAME']).sum().sort_values('SALES', ascending
top_customer = top_customer[['SALES']].round(3)
top_customer.reset_index(inplace = True)
```

In [208…
```
plt.figure(figsize = (15,5))
plt.title('15 Most Valueable Customer (2003 - 2005)', fontsize = 18)
plt.bar(top_customer['CUSTOMERNAME'], top_customer['SALES'], color = '#FF0000', edge
plt.xlabel('Customer Name', fontsize = 15)
plt.ylabel('Revenue', fontsize = 15)
plt.xticks(fontsize = 12, rotation = 90)
plt.yticks(fontsize = 12)
for k, v in top_customer['SALES'].items():
    if v > 600000:
```

```
        plt.text(k, v-270000, '$' + str(v), fontsize = 12, rotation = 90, color = 'b
    else:
        plt.text(k, v+ 50000, '$' + str(v), fontsize = 12, rotation = 90, color = 'b
```



Find out 15 Highest Revenue by Country

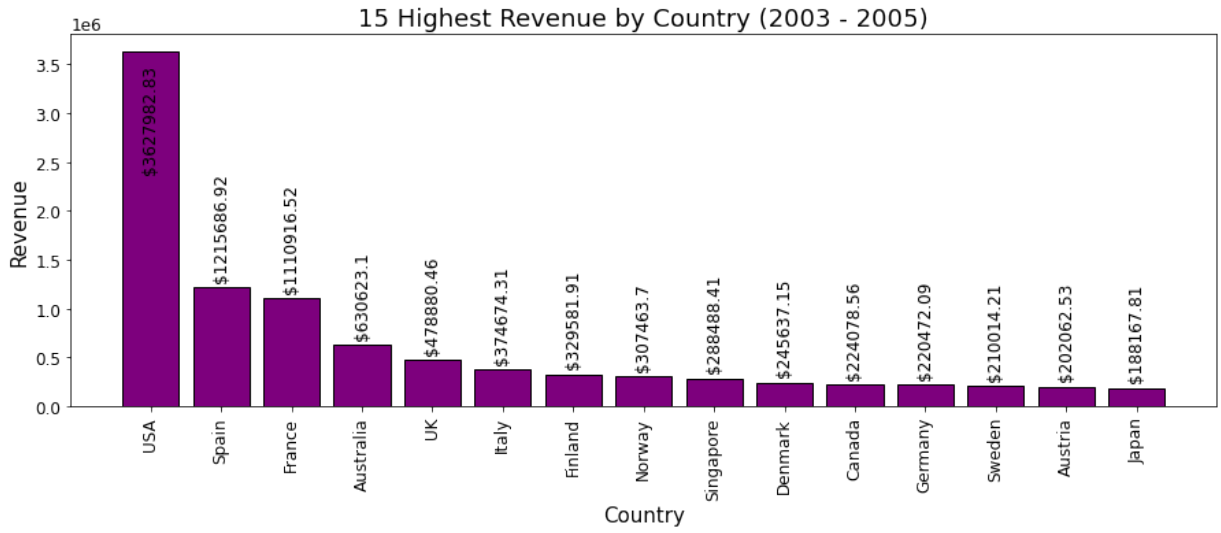
Here are th Top 15 Country which generated the highest revenue

In [203…

```
top_country = sales.groupby(['COUNTRY']).sum().sort_values('SALES', ascending = Fals
top_country = top_country[['SALES']].round(3)
top_country.reset_index(inplace = True)
```

In [232…

```
plt.figure(figsize = (15,5))
plt.title('15 Highest Revenue by Country (2003 - 2005)', fontsize = 18)
plt.bar(top_country['COUNTRY'], top_country['SALES'], color = '#800080', edgecolor =
plt.xlabel('Country', fontsize = 15)
plt.ylabel('Revenue', fontsize = 15)
plt.xticks(fontsize = 12, rotation = 90)
plt.yticks(fontsize = 12)
for k, v in top_country['SALES'].items():
    if v > 3000000:
        plt.text(k, v-1200000, '$' + str(v), fontsize = 12, rotation = 90, color = '
    else:
        plt.text(k, v+100000, '$' + str(v), fontsize = 12, rotation = 90, color = 'b
```



Find out 15 Highest Revenue by City

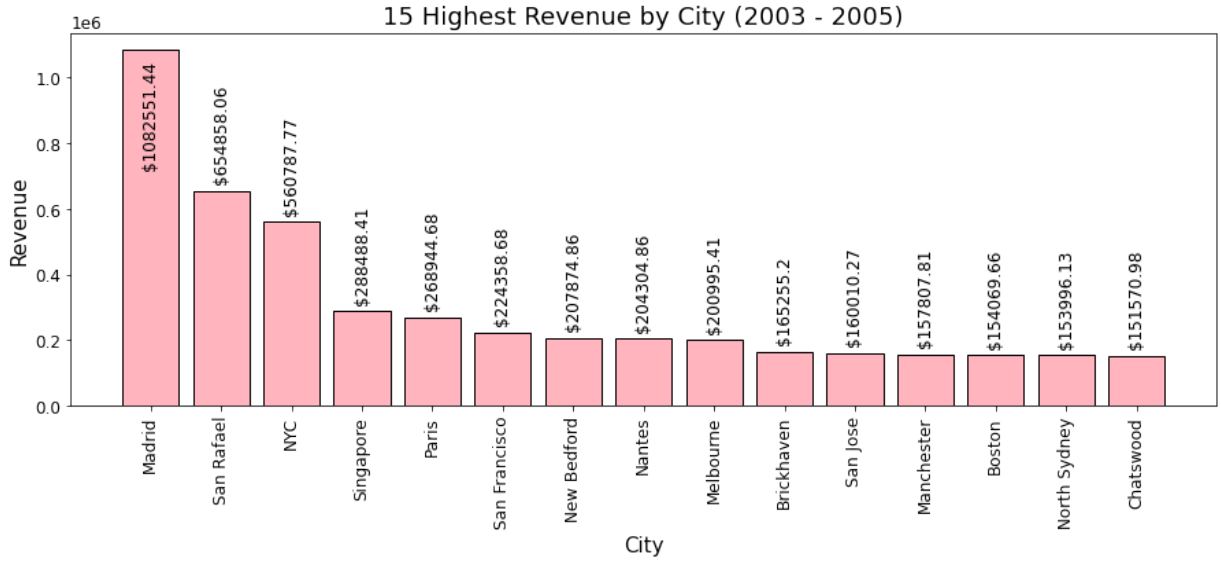
Here are th Top 15 City which generated the highest revenue

In [210…

```
top_city = sales.groupby(['CITY']).sum().sort_values('SALES', ascending = False).hea
top_city = top_city[['SALES']].round(3)
top_city.reset_index(inplace = True)
```

In [231…

```
plt.figure(figsize = (15,5))
plt.title('15 Highest Revenue by City (2003 - 2005)', fontsize = 18)
plt.bar(top_city['CITY'], top_city['SALES'], color = '#FFB6C1', edgecolor = 'black',
plt.xlabel('City', fontsize = 15)
plt.ylabel('Revenue', fontsize = 15)
plt.xticks(fontsize = 12, rotation = 90)
plt.yticks(fontsize = 12)
for k, v, in top_city['SALES'].items():
    if v > 800000:
        plt.text(k, v-350000, '$' + str(v), fontsize = 12, rotation = 90, color = 'b
    else:
        plt.text(k, v+35000, '$' + str(v), fontsize = 12, rotation = 90, color = 'bl
```



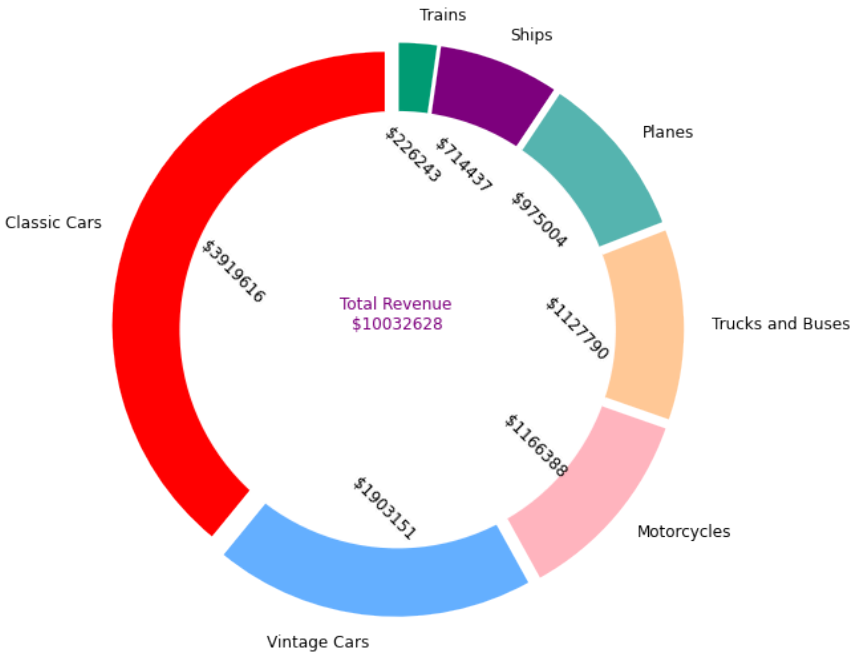
Which products give the highest revenue

In [215…

```
top_product = sales.groupby(['PRODUCTLINE']).sum().sort_values('SALES', ascending =
top_product = top_product[['SALES']]
top_product.reset_index(inplace = True)
total_revenue_product = top_product['SALES'].sum()
total_revenue_product = str(int(total_revenue_product))
total_revenue_product = '$' + total_revenue_product
```

In [230…

```
plt.rcParams['figure.figsize'] = (13,7)
plt.rcParams['font.size'] = 12.0
plt.rcParams['font.weight'] = 6
def format_autopct(values):
    def my_format(pct):
        total = sum(values)
        val = int(round(pct*total/100.0))
        return ' ${v:d}'.format(v = val)
    return my_format
colors = ['#FF0000','#66b3ff','#FFB6C1','#ffcc99','#55B4B0','#800080','#009B77']
explode = (0.05,0.05,0.05,0.05,0.05,0.05,0.05)
fig1, ax1 = plt.subplots()
pie1 = ax1.pie(top_product['SALES'], colors = colors, labels = top_product['PRODUCTL
fraction_text_list = pie1[2]
for text in fraction_text_list:
    text.set_rotation(315)
center_circle = plt.Circle((0,0), 0.80, fc = 'white')
fig = plt.gcf()
fig.gca().add_artist(center_circle)
ax1.axis('equal')
label = ax1.annotate('Total Revenue \n' + str(total_revenue_product), color = '#8000
plt.tight_layout()
plt.show()
```



As depicted in the figure above, Classic Cars contributed the highest revenue, amounting to approximately $3,919,616. The total revenue generated by all product lines summed up to 10,032,628.

Correlation Test

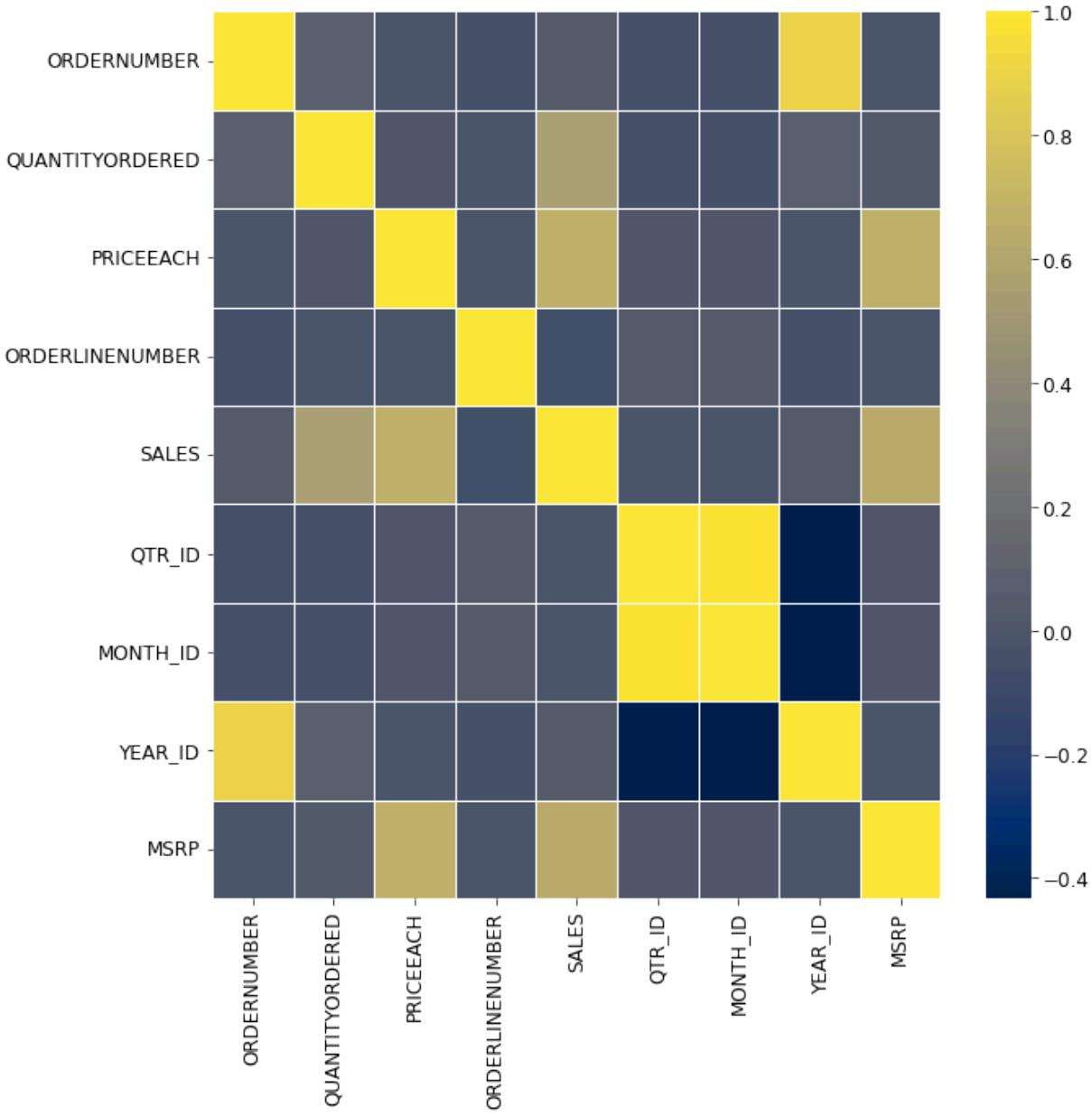
Creating a correlation matrix allows us to visualize the relationships between different features.

In [238…

```
plt.figure(figsize = (10,10))
corr_matrix = sales.corr()
sns.heatmap(corr_matrix, cmap='cividis',linewidths=0.5)
```

Out[238…

```
<AxesSubplot:>
```



Your observations provide insightful insights into the correlation structure of the dataset:

- There is a high positive correlation between ORDERNUMBER and YEAR_ID, indicating that as the order number increases, the year also tends to increase, suggesting a temporal ordering of orders.
- Similarly, a positive correlation is observed between QTR_ID and MONTH_ID, implying that as the quarter number increases, the month also tends to increase, which aligns with the temporal structure of quarters and months.
- SALES, QUANTITYORDERED, PRICEEACH, and MSRP exhibit strong positive correlations, suggesting that these variables tend to increase or decrease together, indicating a strong relationship in sales-related metrics.

- YEAR_ID shows a negative correlation with both QTR_ID and MONTH_ID, suggesting that as the year increases, the quarter and month tend to decrease, which is expected as time progresses sequentially.

These observations provide valuable insights into the relationships between different variables in the dataset, aiding in further analysis and understanding of the underlying patterns.

> > > # Thank you