Sales Data Analysis

Introduction

In this project, a sales data is analyzed using different python packages. The process involves data importation, data cleaning, data transformation, analysis and summary.

Data importation, cleaning and transformation

```
# importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('whitegrid')
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
```

```
# reading the dataset
df = pd.read_csv('sales data.csv')
print('Number of rows: ', df.shape[0])
print('Number of columns: ', df.shape[1])
df.head(5)
```

```
Number of rows:  2823
Number of columns:  25
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE	CUSTOMERNAME	COUNTRY
0	10107	30	95.70	2	2871.00	2/24/2003 0:00:00	Shipped	1.0	2.0	2003.0	Motorcycles	95.70	S10_1678	Land of Toys Inc.	USA
1	10121	34	81.35	5	2765.90	5/7/2003 0:00:00	Shipped	2.0	5.0	2003.0	Motorcycles	95.70	S10_1678	Reims Collectables	France
2	10134	41	94.74	2	3884.34	7/1/2003 0:00:00	Shipped	3.0	7.0	2003.0	Motorcycles	95.70	S10_1678	Lyon Souvenirs	France
3	10145	45	83.26	6	3746.70	8/25/2003 0:00:00	Shipped	3.0	8.0	2003.0	Motorcycles	95.70	S10_1678	Toms Spezialitäten	Germany
4	10168	36	96.00	1	3456.00	10/28/2003 0:00:00	Shipped	4.0	10.0	2003.0	Motorcycles	95.70	S10_1678	Technics Stores Inc.	USA

5 rows × 25 columns

```
# checking the dimension of the data
print('Number of rows: ', df.shape[0])
print('Number of columns: ', df.shape[1])
```

```
Number of rows:  2823
Number of columns:  25
```

```
# checking the data types of the columns
df.dtypes
```

```
ORDERNUMBER       int64
QUANTITYORDERED   int64
PRICEEACH       float64
ORDERLINENUMBER   int64
SALES           float64
ORDERDATE        object
STATUS           object
QTR_ID          float64
MONTH_ID        float64
YEAR_ID         float64
PRODUCTLINE      object
MSRP             int64
PRODUCTCODE      object
CUSTOMERNAME     object
PHONE            object
ADDRESSLINE1     object
ADDRESSLINE2     object
CITY             object
STATE            object
POSTALCODE       object
COUNTRY          object
TERRITORY        object
CONTACTLASTNAME  object
CONTACTFIRSTNAME object
DEALSIZE         object
dtype: object
```

```
# checking the number of null values
df.isnull().sum()
```

```
ORDERNUMBER       0
QUANTITYORDERED   0
PRICEEACH         0
ORDERLINENUMBER   0
SALES             0
ORDERDATE         0
STATUS            0
QTR_ID            0
MONTH_ID          0
YEAR_ID           0
PRODUCTLINE       0
MSRP              0
PRODUCTCODE       0
CUSTOMERNAME      0
PHONE             0
ADDRESSLINE1      0
ADDRESSLINE2     2521
CITY              0
STATE             1486
POSTALCODE        76
COUNTRY           0
TERRITORY         1074
CONTACTLASTNAME   0
CONTACTFIRSTNAME  0
DEALSIZE          0
dtype: object
```

```
# from the above output, the following can be observed:
# 1. the ADDRESSLINE2 column has a lot of missing values and the territory column
# 2. the STATE column has a significant number of null values and hence will be removed from the dataset.
# 3. the POSTALCODE and TERRITORY columns are also missing values.
# 4. the Phone and Contactlastname and contactfirstname columns are also missing values.

# The following can be inferred from the above data:
# 1. There are few null values in the dataset apart from the territory column.
# 2. The territory column has a significant number of null values and hence will be removed from the dataset.
# 3. We will need some name columns for our analysis, hence, they will be removed
# 4. The Phone and Name columns will be unimportant together.

# Dropping the columns that are not important for our analysis
df.drop(['ADDRESSLINE2', 'STATE', 'POSTALCODE', 'TERRITORY', 'PHONE', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME'], axis=1, inplace=True)
df.head(5)
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE	CUSTOMERNAME	CITY	COUNTRY	DEALSIZE
0	10107	30	95.70	2	2871.00	2/24/2003 0:00:00	Shipped	1.0	2.0	2003.0	Motorcycles	95.70	S10_1678	Land of Toys Inc.	NYC	USA	Small
1	10121	34	81.35	5	2765.90	5/7/2003 0:00:00	Shipped	2.0	5.0	2003.0	Motorcycles	95.70	S10_1678	Reims Collectables	Reims	France	Small
2	10134	41	94.74	2	3884.34	7/1/2003 0:00:00	Shipped	3.0	7.0	2003.0	Motorcycles	95.70	S10_1678	Lyon Souvenirs	Paris	France	Medium
3	10145	45	83.26	6	3746.70	8/25/2003 0:00:00	Shipped	3.0	8.0	2003.0	Motorcycles	95.70	S10_1678	Toms Spezialitäten	Pasadena	Germany	Medium
4	10168	36	96.00	1	3456.00	10/28/2003 0:00:00	Shipped	4.0	10.0	2003.0	Motorcycles	95.70	S10_1678	Technics Stores Inc.	San Francisco	USA	Medium

```
# from the above output, the following can be observed:
# 1. the ORDERDATE column is in the object format from the territory column.
# 2. The territory column has a significant number of null values and hence will be removed from the dataset.
# 3. We will need some name columns for our analysis, hence, they will be removed
# 4. The Phone and Name columns will be unimportant together.

# The following can be inferred from the above data:
# 1. There are few null values in the dataset apart from the territory column.
# 2. The territory column has a significant number of null values and hence will be removed from the dataset.
# 3. We will need some name columns for our analysis, hence, they will be removed
# 4. The Phone and Name columns will be unimportant together.

# checking the data types
df['ORDERDATE'].dtypes
```

```
dtype('O')
```

```
# converting the ORDERDATE column to datetime
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df['ORDERDATE'].dtypes
```

```
dtype('O')
```

```
# checking the number of unique values in the ORDERDATE column
df['ORDERDATE'].nunique()
```

```
1265
```

```
# checking the unique values in the ORDERDATE column
df['ORDERDATE'].unique()
```

```
array['2003-02-24 00:00:00', '2003-05-07 00:00:00',
'2003-07-01 00:00:00', '2003-08-25 00:00:00',
'2003-10-28 00:00:00', '2003-11-11 00:00:00',
'2003-12-02 00:00:00', '2004-02-20 00:00:00',
'2004-03-01 00:00:00', '2004-03-10 00:00:00',
'2004-04-19 00:00:00', '2004-05-31 00:00:00',
'2004-06-29 00:00:00', '2004-07-12 00:00:00',
'2004-08-05 00:00:00', '2004-09-08 00:00:00',
'2004-09-24 00:00:00', '2004-10-08 00:00:00',
'2004-10-22 00:00:00', '2004-11-05 00:00:00',
'2004-11-19 00:00:00', '2004-12-03 00:00:00',
'2005-01-06 00:00:00', '2005-01-20 00:00:00',
'2005-02-03 00:00:00', '2005-02-17 00:00:00',
'2005-03-31 00:00:00']
```

```
# checking the dimension of the data
print('Number of rows: ', df.shape[0])
print('Number of columns: ', df.shape[1])
```

```
Number of rows:  2823
Number of columns:  25
```

```
# checking the number of null values
df.isnull().sum()
```

```
ORDERNUMBER       0
QUANTITYORDERED   0
PRICEEACH         0
ORDERLINENUMBER   0
SALES             0
ORDERDATE         0
STATUS            0
QTR_ID            0
MONTH_ID          0
YEAR_ID           0
PRODUCTLINE       0
MSRP              0
PRODUCTCODE       0
CUSTOMERNAME      0
CITY              0
COUNTRY           0
DEALSIZE          0
dtype: object
```

```
# checking the dimension of the dataset
print('Number of rows: ', df.shape[0])
print('Number of columns: ', df.shape[1])
```

```
Number of rows:  2823
Number of columns:  25
```

Analyzing the data using visualization, aggregation and statistics

```
# descriptive summary for the sales
df.describe()
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTR_ID	MONTH_ID	YEAR_ID	MSRP
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000
mean	10258.725115	35.092809	83.658544	6.475883	3553.889072	2.717676	7.092055	2003.815834	100.715551
std	92085.386059	9.741440	20.174277	4.465068	1841.865106	1.203086	3.476468	0.699588	40.187912
min	10100.000000	6.000000	26.88000	1.000000	482.130000	1.000000	1.000000	2003.000000	33.000000
25%	10180.000000	27.000000	68.860000	3.000000	2203.430000	2.000000	5.000000	2003.000000	68.000000
50%	10262.000000	35.000000	95.700000	6.000000	3184.800000	3.000000	7.000000	2004.000000	99.000000
75%	10350.500000	43.000000	100.000000	9.000000	4508.000000	4.000000	11.000000	2004.000000	124.000000
max	10425.000000	97.000000	100.000000	18.000000	14082.800000	4.000000	12.000000	2005.000000	214.000000

```
# checking the data type of the dataset
df.dtypes
```

```
ORDERNUMBER       int64
QUANTITYORDERED   int64
PRICEEACH       float64
ORDERLINENUMBER   int64
SALES           float64
ORDERDATE       datetime64[ns]
STATUS           object
QTR_ID          float64
MONTH_ID        float64
YEAR_ID         float64
PRODUCTLINE      object
MSRP             int64
PRODUCTCODE      object
CUSTOMERNAME     object
CITY             object
COUNTRY          object
DEALSIZE         object
dtype: object
```

```
# checking the unique values in the columns
df.nunique()
```

```
ORDERNUMBER       2823
QUANTITYORDERED     58
PRICEEACH         1981
ORDERLINENUMBER     18
SALES             2763
ORDERDATE         1265
STATUS               9
QTR_ID               4
MONTH_ID            12
YEAR_ID              3
PRODUCTLINE          7
MSRP                80
PRODUCTCODE        109
CUSTOMERNAME        92
CITY                73
COUNTRY             19
DEALSIZE             3
dtype: object
```

```
# checking the dimension of the dataset
print('Number of rows: ', df.shape[0])
print('Number of columns: ', df.shape[1])
```

```
Number of rows:  2823
Number of columns:  25
```

```
# checking the unique values in the STATUS column
print('Number of unique values: ', df['STATUS'].nunique())
print('Unique values: ', df['STATUS'].unique())
```

```
Number of unique values:  9
Unique values:  ['Shipped' 'Disputed' 'In Process' 'Cancelled' 'Resolved' 'On Hold'
 'Order In Hold']
```

```
# checking the unique values in the YEAR_ID column
print('Number of unique values: ', df['YEAR_ID'].nunique())
print('Unique values: ', df['YEAR_ID'].unique())
```

```
Number of unique values:  3
Unique values:  [2003. 2004. 2005.]
```

```
# checking the unique values in the PRODUCTLINE column
print('Number of unique values: ', df['PRODUCTLINE'].nunique())
print('Unique values: ', df['PRODUCTLINE'].unique())
```

```
Number of unique values:  7
Unique values:  ['Motorcycles' 'Classic Cars' 'Trucks and Buses' 'Vintage Cars'
 'Planes' 'Ships' 'Trains']
```

```
# checking the unique values in the COUNTRY column
print('Number of unique values: ', df['COUNTRY'].nunique())
print('Unique values: ', df['COUNTRY'].unique())
```

```
Number of unique values:  19
Unique values:  ['USA' 'France' 'Germany' 'Norway' 'Finland' 'Canada' 'Denmark'
 'Japan' 'Spain' 'Australia' 'Sweden' 'Austria' 'UK' 'Italy' 'Belgium'
 'Philippines' 'Ireland' 'Switzerland' 'Singapore']
```

```
# checking the unique values in the DEALSIZE column
print('Number of unique values: ', df['DEALSIZE'].nunique())
print('Unique values: ', df['DEALSIZE'].unique())
```

```
Number of unique values:  3
Unique values:  ['Small' 'Medium' 'Large']
```

```
# checking the unique values in the PRODUCTCODE column
print('Number of unique values: ', df['PRODUCTCODE'].nunique())
print('Unique values: ', df['PRODUCTCODE'].unique())
```

```
Number of unique values:  109
Unique values:  ['S10_1678' 'S10_1949' 'S10_2016' 'S10_4698' 'S10_4757'
 'S10_4962' 'S12_1099' 'S12_1108' 'S12_3891' 'S12_3990' 'S12_4675'
 'S18_1097' 'S18_1129' 'S18_1589' 'S18_1889' 'S18_2238' 'S18_2319'
 'S18_2325' 'S18_2432' 'S18_2581' 'S18_3136' 'S18_3232' 'S18_3282'
 'S18_3320' 'S18_3856' 'S18_4027' 'S18_4409' 'S18_4522' 'S18_4600'
 'S18_4668' 'S18_5089' 'S18_5092' 'S18_5174' 'S18_5197' 'S18_5232'
 'S18_5346' 'S18_5418' 'S18_5446' 'S18_5571' 'S18_5574' 'S18_5630'
 'S18_5660' 'S18_5697' 'S18_5746' 'S18_5754' 'S18_5768' 'S18_5773'
 'S18_5846' 'S18_5873' 'S18_5879' 'S18_5895' 'S18_5917' 'S18_5949'
 'S18_5958' 'S18_5963' 'S18_5987' 'S18_5994' 'S18_6191' 'S18_6291'
 'S18_6310' 'S18_6322' 'S18_6412' 'S18_6417' 'S18_6463' 'S18_6482'
 'S18_6505' 'S18_6518' 'S18_6665' 'S18_6678' 'S18_6755' 'S18_6773'
 'S18_6787' 'S18_6791' 'S18_6796' 'S18_6874' 'S18_6933' 'S18_6964'
 'S18_7033' 'S18_7089' 'S18_7116' 'S18_7132' 'S18_7234' 'S18_7310'
 'S18_7326' 'S18_7342' 'S18_7405' 'S18_7411' 'S18_7463' 'S18_7481'
 'S18_7693' 'S18_7697' 'S18_7746' 'S18_7755' 'S18_7771' 'S18_7822'
 'S18_7856' 'S18_7933' 'S18_7964' 'S18_7976' 'S18_8294' 'S18_8357'
 'S18_8408' 'S18_8469' 'S18_8544' 'S18_8627' 'S18_8735' 'S18_8746'
 'S18_8748' 'S18_8757' 'S18_8807' 'S18_8856' 'S18_8959' 'S18_9009'
 'S18_9037' 'S18_9074' 'S18_9126' 'S18_9180' 'S18_9232' 'S18_9308'
 'S18_9342' 'S18_9393' 'S18_9417' 'S18_9464' 'S18_9479' 'S18_9514'
 'S18_9582' 'S18_9600' 'S18_9746' 'S18_9797' 'S18_9857' 'S18_9873'
 'S18_9898' 'S18_9908' 'S18_9913' 'S18_9926' 'S24_1046' 'S24_1578'
 'S24_1628' 'S24_1785' 'S24_1937' 'S24_2000' 'S24_2011' 'S24_2022'
 'S24_2300' 'S24_2360' 'S24_2766' 'S24_2768' 'S24_2840' 'S24_2887'
 'S24_3151' 'S24_3191' 'S24_3371' 'S24_3420' 'S24_3432' 'S24_3816'
 'S24_3856' 'S24_3949' 'S24_3969' 'S24_4048' 'S24_4258' 'S24_4278'
 'S24_4620' 'S24_4635' 'S24_4721' 'S24_4773' 'S24_4857' 'S24_4972'
 'S24_5182' 'S24_5476' 'S24_5629' 'S24_5750' 'S24_5804' 'S24_5846'
 'S24_6412' 'S24_6550' 'S24_6563' 'S24_6755' 'S24_6786' 'S24_6804'
 'S24_6809' 'S24_6866' 'S24_6972' 'S24_7041' 'S24_7090' 'S24_7327'
 'S24_7416' 'S24_7418' 'S24_7485' 'S24_7503']
```

```
# checking the dimension of the dataset
print('Number of rows: ', df.shape[0])
print('Number of columns: ', df.shape[1])
```

```
Number of rows:  2823
Number of columns:  25
```

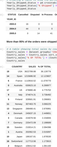
```
# a bar plot showing the count of orders by year
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='YEAR_ID', data=df, palette = 'green')
plt.xlabel('Year')
plt.ylabel('Number of Orders')
plt.show()
```

```
# a bar plot showing the count of orders by qtr
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='QTR_ID', data=df, palette = 'green')
plt.xlabel('Quarters')
plt.ylabel('Number of Orders')
plt.show()
```

```
# a bar plot showing the count of orders by month
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='MONTH_ID', data=df, palette = 'green')
plt.xlabel('Month')
plt.ylabel('Number of Orders')
plt.show()
```

```
# a bar plot showing the count of orders by status
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='STATUS', data=df, palette = 'green')
plt.xlabel('Status')
plt.ylabel('Number of Orders')
plt.show()
```

```
# a bar plot showing the count of orders by deal size
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='DEALSIZE', data=df, palette = 'green')
plt.xlabel('Deal Size')
plt.ylabel('Number of Orders')
plt.show()
```




```
# a bar plot showing the count of orders by product line
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='PRODUCTLINE', data=df, palette = 'green')
plt.xlabel('Product Line')
plt.ylabel('Number of Orders')
plt.show()
```

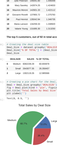
```
# a bar plot showing the sales by year
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x='YEAR_ID', y='SALES', data=df, palette = 'green')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.title('Sales by Year')
plt.show()
```

The sales case increasing from 2003 to 2004 then it drastically went down in 2005.

```
# a bar plot showing the sales by qtr
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x='QTR_ID', y='SALES', data=df, palette = 'green')
plt.xlabel('Quarters')
plt.ylabel('Sales')
plt.title('Sales by Quarters')
plt.show()
```

The quarterly revenue shows that the sales power of the first two years had the highest sales for each year before the crash of the following year.

```
# a bar plot by month
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x='MONTH_ID', y='SALES', data=df, palette = 'green')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales by Month')
plt.show()
```




```
# a bar plot showing the status of the orders for each year
plt.figure(figsize=(6,4), dpi=100)
sns.countplot(x='YEAR_ID', hue='STATUS', data=df, palette = 'green')
plt.xlabel('Year')
plt.ylabel('Number of Orders')
plt.title('Shipment Status by Year')
plt.legend(['Shipped', 'In Process', 'Disputed', 'Cancelled', 'Resolved', 'On Hold', 'Order In Hold'])
plt.show()
```

```
# a bar plot showing the sales of the top 10 countries
df1 = df.groupby('COUNTRY')['SALES'].sum().sort_values(ascending=False)[:10]
df1
```

```
COUNTRY
USA        1036688.00
Spain       389486.00
France      339543.00
Australia   252411.00
UK          190755.00
Italy       174572.00
Finland     132626.00
Norway      131779.00
Singapore   108777.00
Canada      106740.00
Denmark     100811.00
Japan        96503.00
Belgium      79380.00
Sweden       77658.00
Austria      72908.00
Philippines  70616.00
Ireland      57756.00
Switzerland  51544.00
Name: SALES, dtype: float64
```

```
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x=df1.index, y=df1.values, palette = 'green')
plt.xlabel('Country')
plt.ylabel('Sales')
plt.title('Percentage Of Total Sales by Country')
plt.show
```

```
# a bar plot showing the sales of the top 10 customers
df2 = df.groupby('CUSTOMERNAME')['SALES'].sum().sort_values(ascending=False)[:10]
df2
```

```
CUSTOMERNAME
Euro Shopping Channel    715738.98
Mini Gifts Distributors  584188.24
Muscle Machines          591827.34
La Rochelle Gifts        180124.90
Land of Toys Inc.        197736.94
Reims Collectables       176213.96
Lyon Souvenirs           172989.68
Toms Spezialitäten       164069.44
Technics Stores Inc.     158573.12
Daedalus Designs Imports 154622.08
Name: SALES, dtype: float64
```

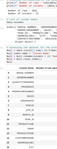
```
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x=df2.index, y=df2.values, palette = 'green')
plt.xlabel('Customer')
plt.ylabel('Sales')
plt.title('Percentage Of Total Sales by Customer')
plt.show
```


Out of 25 orders, Medium has the highest sales of 18% and the rest of the top 10 orders are from USA.

```
# a bar plot showing the sales of the top 10 product lines
df3 = df.groupby('PRODUCTLINE')['SALES'].sum().sort_values(ascending=False)[:10]
df3
```

```
PRODUCTLINE
Classic Cars       1167909.67
Vintage Cars        656627.54
Motorcycles         592001.57
Trucks and Buses    499263.05
Planes              490832.77
Ships               364854.70
Trains              143074.38
Name: SALES, dtyp
```

```
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x=df3.index, y=df3.values, palette = 'green')
plt.xlabel('Product Line')
plt.ylabel('Sales')
plt.title('Percentage Of Total Sales by Product')
plt.show
```



The top 10 products out of 109 accounts for about 30% of total sales.

```
# a bar plot showing the sales of the top 10 product codes
df4 = df.groupby('PRODUCTCODE')['SALES'].sum().sort_values(ascending=False)[:10]
df4
```

```
PRODUCTCODE
S10_1949    214826.35
S18_3232    182091.38
S10_4698    177010.72
S18_2325    131923.91
S18_4600    131210.81
S18_3136    118942.00
S18_3320    117013.39
S12_1099    116083.94
S18_1129    113991.10
S18_1589    112636.43
Name: SALES, dtype: float64
```

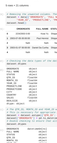
```
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x=df4.index, y=df4.values, palette = 'green')
plt.xlabel('Product Code')
plt.ylabel('Sales')
plt.title('Percentage Of Total Sales by Product')
plt.show
```

Medium size deals accounts for 50% of all sales

```
# a bar plot showing the sales of the top 10 product lines
df5 = df.groupby('DEALSIZE')['SALES'].sum().sort_values(ascending=False)[:10]
df5
```

```
DEALSIZE
Medium    1384620.53
Large     1251991.19
Small     917428.35
Name: SALES, dtype: float64
```

```
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x=df5.index, y=df5.values, palette = 'green')
plt.xlabel('Deal Size')
plt.ylabel('Sales')
plt.title('Percentage Of Total Sales by Deal Size')
plt.show
```



Medium size deals accounts for 50% of all sales

```
# a bar plot showing the sales of the top 10 product lines
df6 = df.groupby('PRODUCTLINE')['SALES'].sum().sort_values(ascending=False)[:10]
df6
```

```
PRODUCTLINE
Classic Cars       1167909.67
Vintage Cars        656627.54
Motorcycles         592001.57
Trucks and Buses    499263.05
Planes              490832.77
Ships               364854.70
Trains              143074.38
Name: SALES, dtyp
```

```
plt.figure(figsize=(6,4), dpi=100)
sns.barplot(x=df6.index, y=df6.values, palette = 'green')
plt.xlabel('Product Line')
plt.ylabel('Sales')
plt.title('Percentage Of Total Sales by Product')
plt.show
```

Summary

The average sales per customer is $3553 million while the minimum and maximum are 482 and 14082 million respectively.

The sales case increasing from 2003 to 2004 then it drastically went down in 2005. The quarterly revenue shows that the first two quarters of the first two years had the highest sales for each year before the crash of the following year.

Out of 25 orders, Medium has the highest sales of 18% and the rest of the top 10 orders are from USA.

The top 10 products out of 109 accounts for about 30% of total sales.

Medium size deals accounts for 50% of all sales.

Recommendations

There was a decline in knowledge into 2004 and 2005 which is a cause of concern.

The company should think of ways to improve sales in other or countries apart from the top 5.

The organization should also investigate why Grand Prix and Gabriel Nelson has crossed as much in the company's products. This might help in advancing and improving the interest of other customers.