Sales Data Analysis

May 24, 2024

 # SALES DATA ANALYSIS with Pandas by Kalyan Neelam

Sales analysis is the process of examining and evaluating sales data to gain insights into the performance of a company's products or services. It involves analyzing various aspects of sales, such as trends, patterns, customer behavior, and performance metrics, to make informed business decisions.

Sales analysis helps companies understand their sales performance, identify areas of strength and weakness, pinpoint opportunities for growth, and develop strategies to improve sales effectiveness.

Key components of sales analysis typically include:

1. **Sales Volume**: Examining the quantity of products or services sold over a specific period.
2. **Revenue Analysis**: Assessing the revenue generated from sales, including total revenue, revenue by product or service, revenue by customer segment, etc.
3. **Sales Trends**: Identifying patterns and trends in sales data over time, such as seasonal fluctuations, cyclical trends, or changes in demand.
4. **Customer Analysis**: Understanding the demographics, preferences, buying behavior, and purchasing patterns of customers to target marketing efforts more effectively and improve customer satisfaction.
5. **Product Performance**: Evaluating the performance of individual products or product categories in terms of sales volume, revenue, profitability, and market share.
6. **Sales Channels**: Analyzing the effectiveness of different sales channels (e.g., direct sales, online sales, distribution channels) and optimizing their performance.
7. **Sales Forecasting**: Using historical sales data and predictive analytics to forecast future sales volumes and revenue.
8. **Competitive Analysis**: Comparing the company's sales performance with that of competitors to identify strengths, weaknesses, and opportunities in the market.

- Store sales and profit analysis help businesses identify areas for improvement and make data-driven decisions to optimize their operations, pricing, marketing, and inventory management strategies to drive revenue and growth.

0.1 Pandas

Pandas is a powerful Python library for data manipulation and analysis, and it plays a crucial role in sales analysis.

1. **Data Cleaning and Preprocessing**: Pandas provides functions for handling missing data, removing duplicates, and transforming data. We can use pandas to clean and preprocess sales data, ensuring that it is accurate and consistent before analysis.
2. **Data Manipulation**: Pandas offers powerful tools for data manipulation, such as filtering, sorting, grouping, and aggregating data. We can use pandas to perform calculations, calculate summary statistics, and reshape data to extract meaningful insights from sales data.
3. **Time Series Analysis**: Pandas has built-in support for time series data, making it easy to analyze sales data over time. We can use pandas to resample time series data, calculate rolling statistics, and perform date/time-based operations to understand sales trends and patterns.
4. **Data Visualization Integration**: Pandas seamlessly integrates with data visualization libraries like Matplotlib and Seaborn, allowing to create insightful visualizations of sales data.
5. **Data Merging and Joining**: Pandas provides functions for merging and joining multiple datasets based on common keys or indices. This capability allows to combine sales data with other relevant datasets, such as customer data or product data, to perform more comprehensive analysis and gain deeper insights into sales performance.

0.2 Case study 1: SuperStore Sales Analysis

0.3 OBJECTIVE

- What is the overall sales trend?
- Sales by Category?
- Sales by Sub-Category?
- Profit Analysis
- Profit analysis by customer segments:
- Which are the Top 10 products by sales?
- Which are the Most Selling Products?
- Which is the most preferred Ship Mode?
- Which are the Most Profitable Category and Sub-Category?

IMPORTING REQUIRED LIBRARIES

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

[16]:
```
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as colors
```

```
pio.templates.default = "plotly_white"
```

LOADING THE DATASET

[5]:
```
# Read Excel file into a pandas DataFrame
df = pd.read_excel("/content/superstore_sales.xlsx")

# Display the DataFrame
df.head()
```

[5] :

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	region	...	category	sub_category	product_name	sales	quantity	discount	profit	shipping_cost	order_priority	year
0	AG-2011-2040	2011-01-01	2011-01-06	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria	Africa	Africa	...	Office Supplies	Storage	Tenex Lockers, Blue	408.300	2	0.0	106.140	35.46	Medium	2011
1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	Oceania	...	Office Supplies	Supplies	Acme Trimmer, High Speed	120.366	3	0.1	36.036	9.72	Medium	2011
2	HU-2011-1220	2011-01-01	2011-01-05	Second Class	Annie Thurman	Consumer	Budapest	Hungary	EMEA	EMEA	...	Office Supplies	Storage	Tenex Box, Single Width	66.120	4	0.0	29.640	8.17	High	2011
3	IT-2011-3647632	2011-01-01	2011-01-05	Second Class	Eugene Moren	Home Office	Stockholm	Sweden	EU	North	...	Office Supplies	Paper	Enermax Note Cards, Premium	44.865	3	0.5	-26.055	4.82	High	2011
4	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	Oceania	...	Furniture	Furnishings	Eldon Light Bulb, Duo Pack	113.670	5	0.1	37.770	4.70	Medium	2011

[5 rows x 21 columns]

[6] :
```
# Last five rows of the dataset
df.tail()
```

[6]:

	order_id	order_date	ship_date	ship_mode	customer_name
51285	CA-2014-115427	2014-12-31	2015-01-04	Standard Class	Erica Bern
51286	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	Liz Preis
51287	MX-2014-110527	2014-12-31	2015-01-02	Second Class	Charlotte Melton

```
51288  MX-2014-114783  2014-12-31  2015-01-06  Standard Class      Tamara Dahlen
51289  CA-2014-156720  2014-12-31  2015-01-04  Standard Class      Jill Matthias

        segment              state        country market   region  ... \
51285  Corporate         California  United States     US     West  ...
51286  Consumer  Souss-Massa-Draâ           Morocco Africa   Africa  ...
51287  Consumer            Managua      Nicaragua  LATAM  Central  ...
51288  Consumer          Chihuahua         Mexico  LATAM    North  ...
51289  Consumer           Colorado  United States     US     West  ...

            category sub_category  \
51285  Office  Supplies       Binders
51286  Office  Supplies       Binders
51287  Office  Supplies        Labels
51288  Office  Supplies        Labels
51289  Office  Supplies     Fasteners

                                                product_name    sales  quantity  \
51285  Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl   13.904         2
51286          Wilson Jones Hole Reinforcements, Clear    3.990         1
51287           Hon Color Coded Labels, 5000 Label Set   26.400         3
51288           Hon Legal Exhibit Labels, Alphabetical    7.120         1
51289                                Bagged Rubber Bands    3.024         3

       discount    profit  shipping_cost order_priority  year
51285       0.2    4.5188          0.890         Medium  2014
51286       0.0    0.4200          0.490         Medium  2014
51287       0.0   12.3600          0.350         Medium  2014
51288       0.0    0.5600          0.199         Medium  2014
51289       0.2   -0.6048          0.170         Medium  2014

[5 rows x 21 columns]
```

[7] :
```
# Shape of the dataset
df.shape
```

[7]: (51290, 21)

[8] :
```
# Columns present in the dataset
df.columns
```

[8] :
```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year'],
      dtype='object')
```

[9] :
```
# A concise summary of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   order_id        51290 non-null  object
 1   order_date      51290 non-null  datetime64[ns]
 2   ship_date       51290 non-null  datetime64[ns]
 3   ship_mode       51290 non-null  object
 4   customer_name   51290 non-null  object
 5   segment         51290 non-null  object
 6   state           51290 non-null  object
 7   country         51290 non-null  object
 8   market          51290 non-null  object
 9   region          51290 non-null  object
 10  product_id      51290 non-null  object
 11  category        51290 non-null  object
 12  sub_category    51290 non-null  object
 13  product_name    51290 non-null  object
 14  sales           51290 non-null  float64
 15  quantity        51290 non-null  int64
 16  discount        51290 non-null  float64
 17  profit          51290 non-null  float64
 18  shipping_cost   51290 non-null  float64
 19  order_priority  51290 non-null  object
 20  year            51290 non-null  int64
dtypes: datetime64[ns](2), float64(4), int64(2), object(13)
memory usage: 8.2+ MB
```

[10] :
```
# Checking missing values
df.isna().sum()
```

[10] :
```
order_id         0
order_date       0
ship_date        0
ship_mode        0
customer_name    0
segment          0
state            0
country          0
market           0
region           0
product_id       0
category         0
```

```
sub_category      0
product_name      0
sales             0
quantity          0
discount          0
profit            0
shipping_cost     0
order_priority    0
year              0
dtype: int64
```

[11]:
```
# Generating descriptive statistics summary
df.describe().round()
```

[11]:

	order_date	ship_date	sales	\
count	51290	51290	51290.0	
mean	2013-05-11 21:26:49.155780864	2013-05-15 20:42:42.745174528	246.0	
min	2011-01-01 00:00:00	2011-01-03 00:00:00	0.0	
25%	2012-06-19 00:00:00	2012-06-23 00:00:00	31.0	
50%	2013-07-08 00:00:00	2013-07-12 00:00:00	85.0	
75%	2014-05-22 00:00:00	2014-05-26 00:00:00	251.0	
max	2014-12-31 00:00:00	2015-01-07 00:00:00	22638.0	
std	NaN	NaN	488.0	

	quantity	discount	profit	shipping_cost	year
count	51290.0	51290.0	51290.0	51290.0	51290.0
mean	3.0	0.0	29.0	26.0	2013.0
min	1.0	0.0	-6600.0	0.0	2011.0
25%	2.0	0.0	0.0	3.0	2012.0
50%	3.0	0.0	9.0	8.0	2013.0
75%	5.0	0.0	37.0	24.0	2014.0
max	14.0	1.0	8400.0	934.0	2014.0
std	2.0	0.0	174.0	57.0	1.0

0.4 EXPLORATORY DATA ANALYSIS

1. WHAT IS THE OVERALL SALES TREND?

[12]:
```
# Getting month year from order_date
df['month_year'] = df['order_date'].apply(lambda x: x.strftime('%Y-%m'))
```

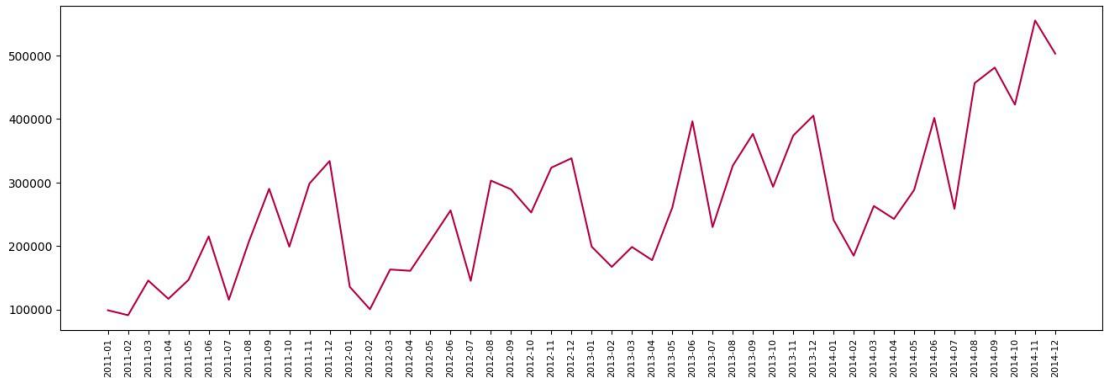
[13]:
```
print(df['month_year'].unique())
print(df['month_year'].dtype)
```

```
['2011-01' '2011-02' '2011-03' '2011-04' '2011-05' '2011-06' '2011-07'
 '2011-08' '2011-09' '2011-10' '2011-11' '2011-12' '2012-01' '2012-02'
 '2012-03' '2012-04' '2012-05' '2012-06' '2012-07' '2012-08' '2012-09'
 '2012-10' '2012-11' '2012-12' '2013-01' '2013-02' '2013-03' '2013-04'
```

```
'2013-05' '2013-06' '2013-07' '2013-08' '2013-09' '2013-10' '2013-11'
 '2013-12' '2014-01' '2014-02' '2014-03' '2014-04' '2014-05' '2014-06'
 '2014-07' '2014-08' '2014-09' '2014-10' '2014-11' '2014-12']
object
```

[14] :
```
# Group by 'month_year' and sum 'sales'
df_temp = df.groupby('month_year')['sales'].sum().reset_index()
```

[15] :
```
# Setting the figure size
plt.figure(figsize=(16, 5))
plt.plot(df_temp['month_year'], df_temp['sales'], color='#b80045')
plt.xticks(rotation='vertical', size=8)
plt.show()
```



- Highest Sales in the month of November in 2014

2. SALES BY CATEGORY

[17]:
```
df.columns
```

[17]:
```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year',
       'month_year'],
      dtype='object')
```

[19]:
```
sales_by_category = df.groupby('category')['sales'].sum().reset_index()


fig = px.pie(sales_by_category,
             values='sales',
             names='category',
             hole=0.5,
```

```
                color_discrete_sequence=px.colors.qualitative.Pastel)

fig.update_traces(textposition='inside',    textinfo='percent+label')
fig.update_layout(title_text='Sales Analysis by Category',␣
 ↪title_font=dict(size=24))

fig.show()
```

- Technology has highest sale.

3. SALES BY SUB CATEGORY

[20]:
```
sales_by_subcategory = df.groupby('sub_category')['sales'].sum().reset_index()
fig = px.bar(sales_by_subcategory,
             x='sub_category',
             y='sales',
             title='Sales Analysis by Sub-Category')
fig.show()
```

- Phones has highest sales

4. MONTHLY PROFITS

[22]:
```
profit_by_month = df.groupby('month_year')['profit'].sum().reset_index()
fig = px.line(profit_by_month, x='month_year', y='profit', title='Monthly␣
 ↪Profit Analysis')
fig.show()
```

5. PROFIT BY CATEGORY

[23]:
```
profit_by_category    =    df.groupby('category')['profit'].sum().reset_index()

fig = px.pie(profit_by_category,
             values='profit',
             names='category',
             hole=0.5,
             color_discrete_sequence=px.colors.qualitative.Pastel)

fig.update_traces(textposition='inside',    textinfo='percent+label')
fig.update_layout(title_text='Profit Analysis by Category',␣
 ↪title_font=dict(size=24))

fig.show()
```

- Technology has highest profit.

6. PROFIT BY SUB-CATEGORY

[25]:
```
profit_by_subcategory = df.groupby('sub_category')['profit'].sum().reset_index()
fig = px.bar(profit_by_subcategory, x='sub_category',
```

```
            y='profit',
            title='Profit Analysis by Sub-Category')
fig.show()
```

- Copiers has highest profit.

7. PROFIT ANALYSIS BY CUSTOMER SEGMENTS

[26]:
```
sales_profit_by_segment = df.groupby('segment').agg({'sales': 'sum', 'profit': 'sum'}).reset_index()

color_palette = colors.qualitative.Pastel

fig = go.Figure()
fig.add_trace(go.Bar(x=sales_profit_by_segment['segment'],
                     y=sales_profit_by_segment['sales'],
                     name='Sales',
                     marker_color=color_palette[0]))
fig.add_trace(go.Bar(x=sales_profit_by_segment['segment'],
                     y=sales_profit_by_segment['profit'],
                     name='Profit',
                     marker_color=color_palette[1]))

fig.update_layout(title='Sales and Profit Analysis by Customer Segment',
                  xaxis_title='Customer Segment', yaxis_title='Amount')

fig.show()
```

8. SALES TO PROFIT RATIO

[28]:
```
sales_profit_by_segment = df.groupby('segment').agg({'sales': 'sum', 'profit': 'sum'}).reset_index()
sales_profit_by_segment['Sales_to_Profit_Ratio'] = sales_profit_by_segment['sales'] / sales_profit_by_segment['profit']
print(sales_profit_by_segment[['segment', 'Sales_to_Profit_Ratio']])
```

```
        segment  Sales_to_Profit_Ratio
0      Consumer               8.686070
1     Corporate               8.637804
2   Home Office               8.338550
```

- The store has higher profits from the product sales for consumers.

9. WHICH ARE THE TOP 10 PRODUCTS BY SALES?

[29]:
```
# Grouping products by sales
prod_sales = pd.DataFrame(df.groupby('product_name')['sales'].sum())

# Sorting the dataframe in descending order
prod_sales.sort_values(by=['sales'], inplace=True, ascending=False)
```

```
# Top 10 products by sales
prod_sales[:10]
```

[29] :

product_name	sales
Apple Smart Phone, Full Size	86935.7786
Cisco Smart Phone, Full Size	76441.5306
Motorola Smart Phone, Full Size	73156.3030
Nokia Smart Phone, Full Size	71904.5555
Canon imageCLASS 2200 Advanced Copier	61599.8240
Hon Executive Leather Armchair, Adjustable	58193.4841
Office Star Executive Leather Armchair, Adjustable	50661.6840
Harbour Creations Executive Leather Armchair, A...	50121.5160
Samsung Smart Phone, Cordless	48653.4600
Nokia Smart Phone, with Caller ID	47877.7857

- Apple smart phone is the top product by sale.

10. WHICH ARE THE MOST SELLING PRODUCTS?

[30] :
```
# Grouping products by Quantity
best_selling_prods = pd.DataFrame(df.groupby('product_name')['quantity'].sum())

# Sorting the dataframe in descending order
best_selling_prods.sort_values(by=['quantity'], inplace=True, ascending=False)

# Most selling products
best_selling_prods[:10]
```

[30]:

product_name	quantity
Staples	876
Cardinal Index Tab, Clear	337
Eldon File Cart, Single Width	321
Rogers File Cart, Single Width	262
Sanford Pencil Sharpener, Water Color	259
Stockwell Paper Clips, Assorted Sizes	253
Avery Index Tab, Clear	252
Ibico Index Tab, Clear	251
Smead File Cart, Single Width	250
Stanley Pencil Sharpener, Water Color	242

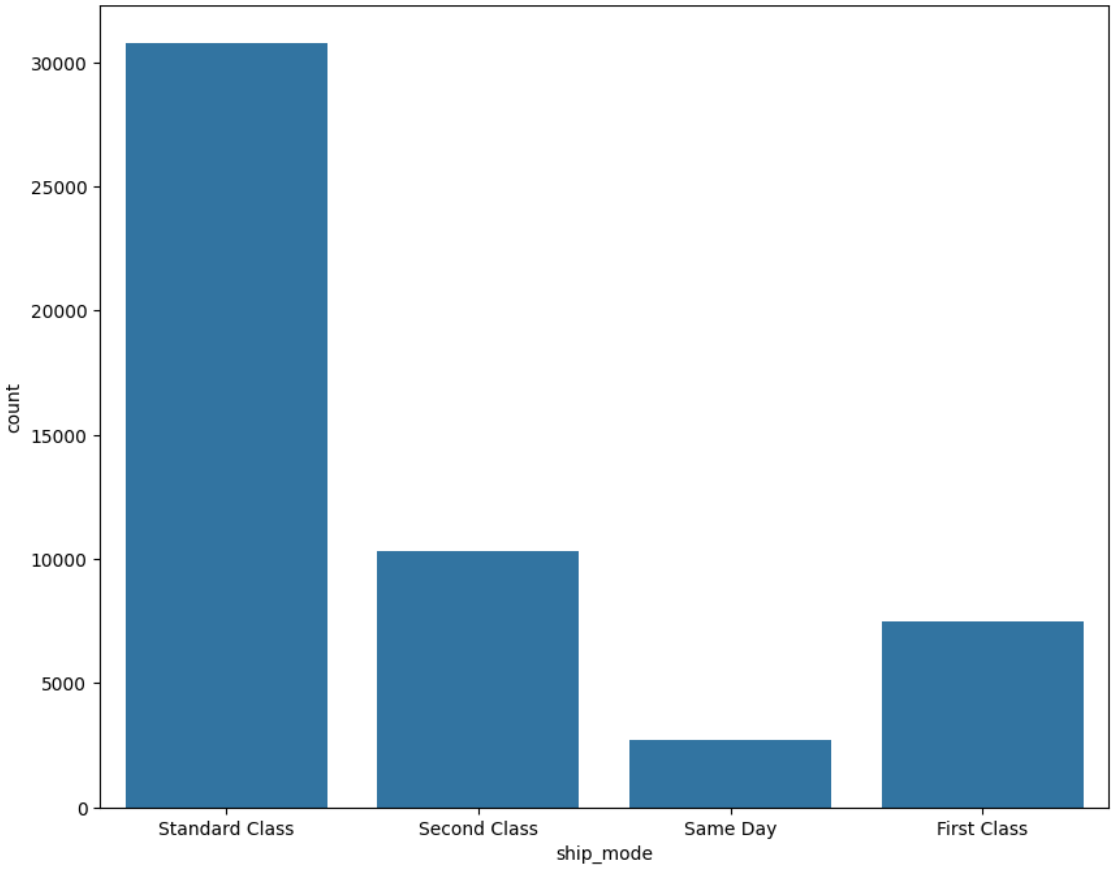
- Staples is the best selling product.

11. WHAT IS THE MOST PREFERRED SHIP MODE?

[ ]:
```
# Setting the figure size
plt.figure(figsize=(10, 8))
```

```
# countplot: Show the counts of observations in each categorical bin using bars
sns.countplot(x='ship_mode', data=df)

# Display the figure
plt.show()
```



- standard class is the most preferred ship mode.

12. WHICH ARE THE MOST PROFITABLE CATEGORY AND SUB-CATEGORY?

[31] :
```
# Grouping products by Category and Sub-Category
cat_subcat = pd.DataFrame(df.groupby(['category', 'sub_category'])['profit'].
 sum())

# Sorting the values
cat_subcat.sort_values(['category','profit'], ascending=False)
```

[31]:

		profit
category	sub_category	

Technology	Copiers	258567.54818
	Phones	216717.00580
	Accessories	129626.30620
	Machines	58867.87300
Office Supplies	Appliances	141680.58940
	Storage	108461.48980
	Binders	72449.84600
	Paper	59207.68270
	Art	57953.91090
	Envelopes	29601.11630
	Supplies	22583.26310
	Labels	15010.51200
	Fasteners	11525.42410
Furniture	Bookcases	161924.41950
	Chairs	141973.79750
	Furnishings	46967.42550
	Tables	-64083.38870

- Technology and copiers

0.5 Case Study 2: To analyze and answer business questions about 12 months worth of sales data.

The data contains hundreds of thousands of electronics store purchases broken down by month, product type, cost, purchase address, etc.

[35]:
```
try:
    # Attempt to read CSV file into a pandas DataFrame
    all_data = pd.read_csv("/content/all_data.csv", encoding='utf-8')
    print("CSV file successfully loaded.")
except Exception as e:
    print("An error occurred while reading the CSV file:", e)
    # Handle the error, or provide appropriate feedback to the user
```

CSV file successfully loaded.

[36]:
```
all_data.head()
```

[36] :

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
1	NaN	NaN	NaN	NaN	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600	
4	176560	Wired Headphones	1	11.99	

	Order Date	Purchase Address
0	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN

```
2  04/07/19 22:30      682 Chestnut St, Boston, MA 02215
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
```

0.6 Drop rows of NAN

[37] :
```
# Find NAN
nan_df = all_data[all_data.isna().any(axis=1)]
display(nan_df.head())
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1	NaN	NaN	NaN	NaN	NaN	NaN
356	NaN	NaN	NaN	NaN	NaN	NaN
735	NaN	NaN	NaN	NaN	NaN	NaN
1433	NaN	NaN	NaN	NaN	NaN	NaN
1553	NaN	NaN	NaN	NaN	NaN	NaN

[38] :
```
all_data = all_data.dropna()
all_data.head()
```

[38]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

	Order Date	Purchase Address
0	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

[39] :
```
all_data = all_data[all_data['Order Date'].str[0:2]!='Or']
```

Make columns correct type

[40] :
```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 185950 entries, 0 to 186849
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          185950 non-null  object
 1   Product           185950 non-null  object
```

```
 2   Quantity Ordered  185950 non-null   object
 3   Price Each        185950 non-null   object
 4   Order Date        185950 non-null   object
 5   Purchase Address  185950 non-null object
dtypes: object(6)
memory usage: 9.9+ MB
```

[41] :
```
all_data.shape
```

[41]: (185950, 6)

[42] :
```
all_data.describe()
```

[42]:

	Order ID	Product	Quantity Ordered	Price Each	\
count	185950	185950	185950	185950	
unique	178437	19	9	23	
top	160873	USB-C Charging Cable	1	11.95	
freq	5	21903	168552	21903	

	Order Date	Purchase Address
count	185950	185950
unique	142395	140787
top	12/15/19 20:16	193 Forest St,San Francisco, CA 94016
freq	8	9

[43] :
```
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
```

Add month column

[44] :
```
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
```

[44]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

	Order Date	Purchase Address	Month
0	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

[45] :
```
all_data['Month'] = pd.to_datetime(all_data['Order Date']).dt.month
all_data.head()
```

```
<ipython-input-45-9ae23976486e>:1: UserWarning:

Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
```

[45]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

	Order Date	Purchase Address	Month
0	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

Add city column

[48]:
```
def get_city(address):
    return address.split(",")[1].strip(" ")

def get_state(address):
    return address.split(",")[2].split(" ")[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x: f"{get_city(x)}
 s ({get_state(x)})")
all_data.head()
```

[48]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

	Order Date	Purchase Address	Month	\
0	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	

```
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4
5  04/30/19 09:27      333 8th St, Los Angeles, CA 90001      4

                City
0         Dallas  (TX)
2         Boston  (MA)
3    Los Angeles  (CA)
4    Los Angeles  (CA)
5    Los Angeles  (CA)
```

Question 1: What was the best month for sales? How much was earned that month?

[49]:
```
# Perform the calculation
all_data['Sales'] = all_data['Quantity Ordered'].astype('int') *␣
 ↪all_data['Price Each'].astype('float')
```

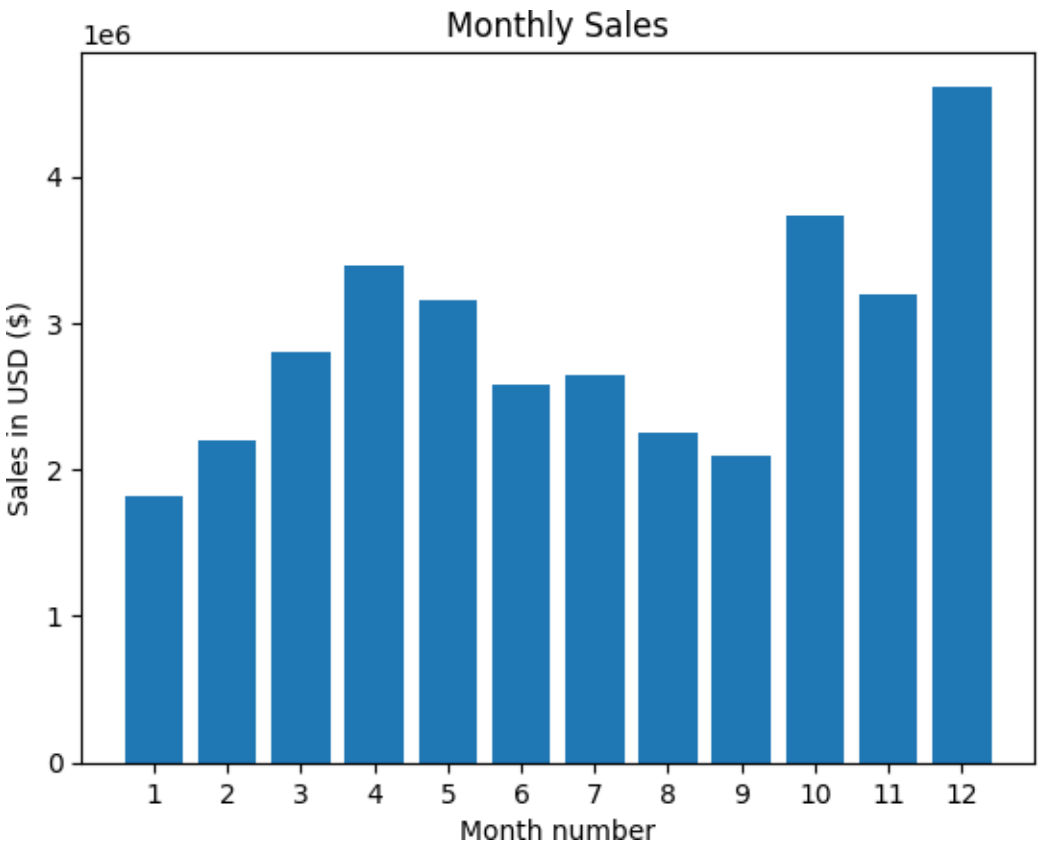
[50]:
```
# Group by 'Month' and sum the numeric columns
monthly_sales = all_data.groupby('Month').agg({'Quantity Ordered': 'sum',␣
 ↪'Price Each': 'sum', 'Sales': 'sum'})
```

[51]:
```
monthly_sales.head()
```

[51]:

Month	Quantity Ordered	Price Each	Sales
1	10903	1811768.38	1822256.73
2	13449	2188884.72	2202022.42
3	17005	2791207.83	2807100.38
4	20558	3367671.02	3390670.24
5	18667	3135125.13	3152606.75

[53]:
```
months = range(1,13)
plt.bar(months,all_data.groupby(['Month'])['Sales'].sum())
plt.xticks(months)
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month number')
plt.title('Monthly Sales')
plt.show()
```



- Month 12 (December) is the highest sales in 2019 with approximately $4,810,000.

Question 2: What city sold the most product?

[54]:
```
city_sales = all_data.groupby('City').agg({'Quantity Ordered': 'sum', 'Price Each': 'sum', 'Sales': 'sum'})
city_sales.head()
```

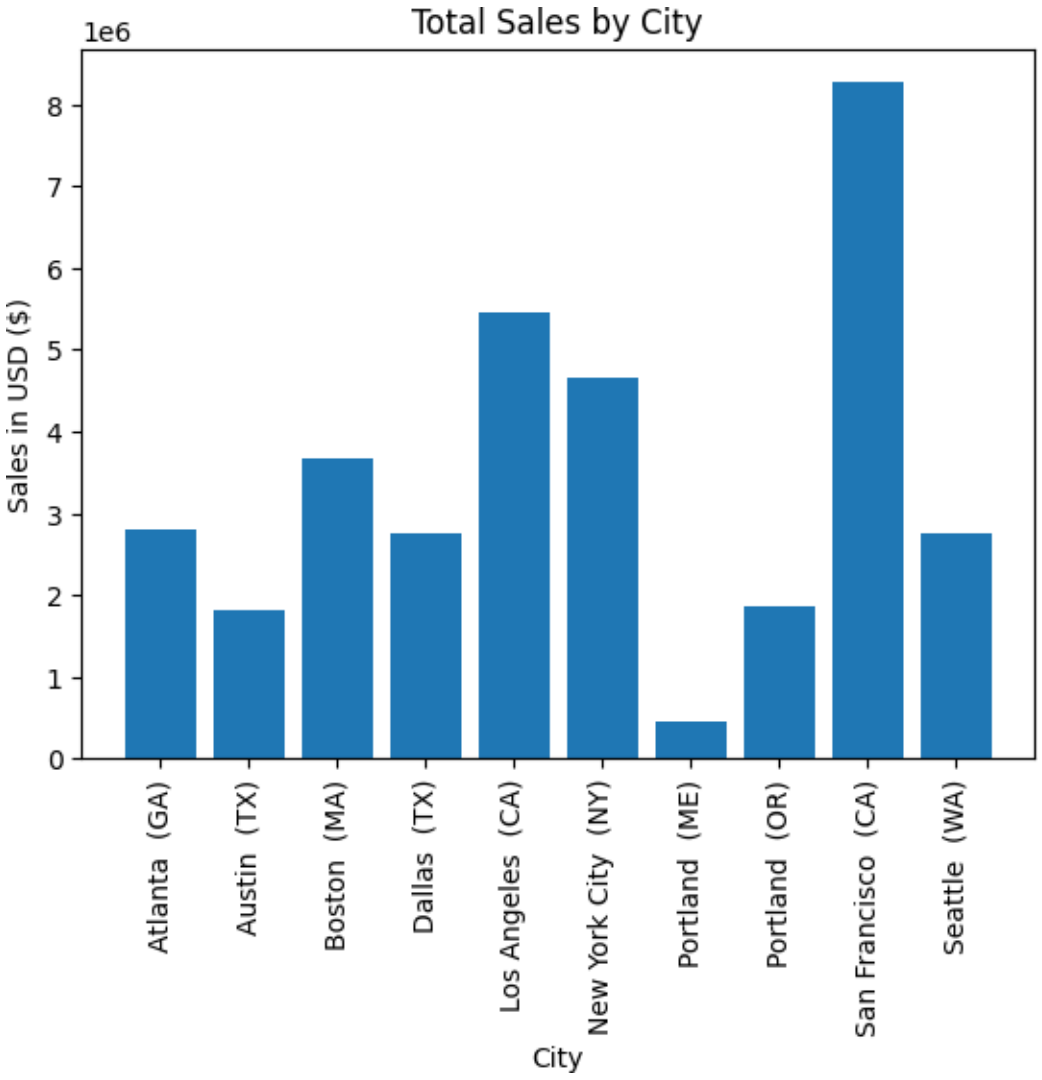
[54]:

City	Quantity Ordered	Price Each	Sales
Atlanta (GA)	16602	2779908.20	2795498.58
Austin (TX)	11153	1809873.61	1819581.75
Boston (MA)	22528	3637409.77	3661642.01
Dallas (TX)	16730	2752627.82	2767975.40
Los Angeles (CA)	33289	5421435.23	5452570.80

[56]:
```
# Calculate monthly sales
city_sales = all_data.groupby('City')['Sales'].sum()

# Plotting
plt.bar(city_sales.index, city_sales.values)
```

```
plt.xticks(rotation=90)
plt.ylabel('Sales in USD ($)')
plt.xlabel('City')
plt.title('Total Sales by City')
plt.show()
```



- San Francisco has highest sales.

Question 3: What time should we display advertisements to maximize likelihood of customer's buying product?

[57]:
```
# Add hour column
all_data['Hour'] = pd.to_datetime(all_data['Order Date']).dt.hour
all_data['Minute'] = pd.to_datetime(all_data['Order Date']).dt.minute
```

```
all_data['Count'] = 1
all_data.head()
```

```
<ipython-input-57-3f3d5aef9003>:2: UserWarning:

Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.

<ipython-input-57-3f3d5aef9003>:3: UserWarning:

Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
```

[57]:

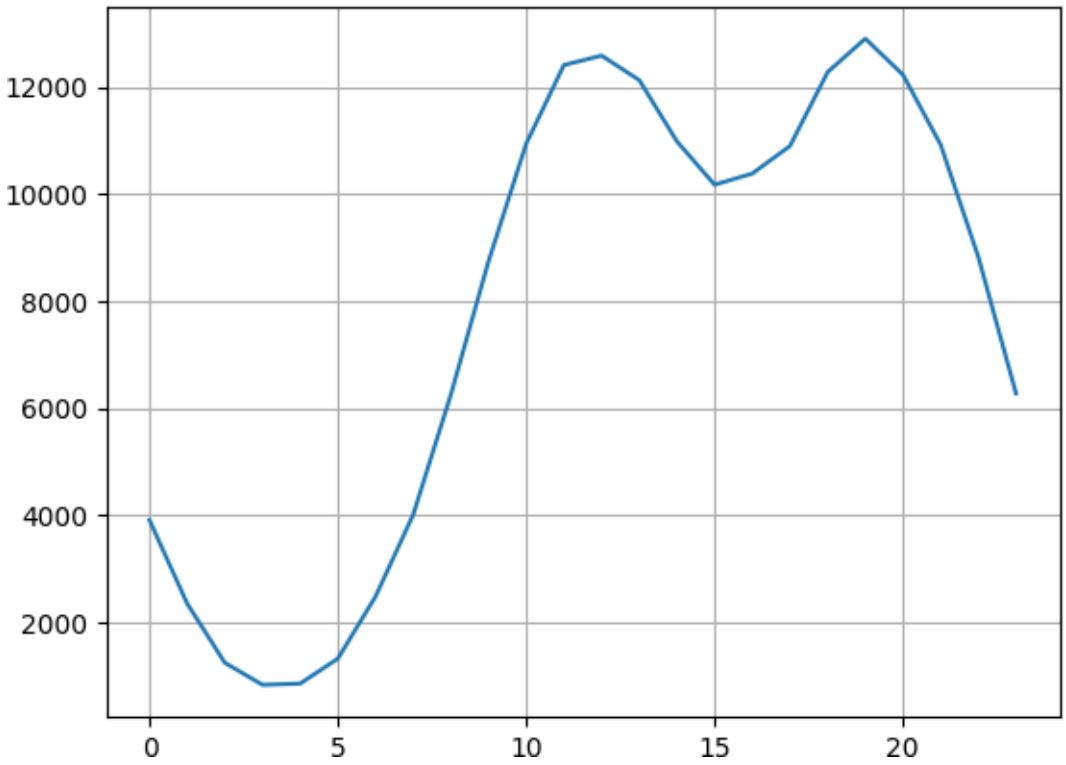
	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

	Order Date	Purchase Address	Month	\
0	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	
2	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	
3	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	
4	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	
5	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	

	City	Sales	Hour	Minute	Count
0	Dallas (TX)	23.90	8	46	1
2	Boston (MA)	99.99	22	30	1
3	Los Angeles (CA)	600.00	14	38	1
4	Los Angeles (CA)	11.99	14	38	1
5	Los Angeles (CA)	11.99	9	27	1

[59]:
```
keys = [pair for pair, df in all_data.groupby(['Hour'])]

plt.plot(keys, all_data.groupby(['Hour']).count()['Count'])
plt.grid()
plt.show()
```



There are approximately 2 peaks at the data. They are 12 (12 PM) and 19 (7 PM). It makes sense since most people shop during the day. From this data, It can suggest to advertise their product right before 12 PM and/or 7 PM. It could be 11.30 AM and/or 6.30 PM.

Question 4: What products are most often sold together?

[60]:
```
df = all_data[all_data['Order ID'].duplicated(keep=False)]
```

[61]:
```
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.
  join(x))
df2 = df[['Order ID', 'Grouped']].drop_duplicates()
```

<ipython-input-61-91e38189159a>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

[62]:
```
from itertools import combinations
from collections import Counter

count = Counter()

for row in df2['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

for key,value in count.most_common(10):
    print(key, value)
```

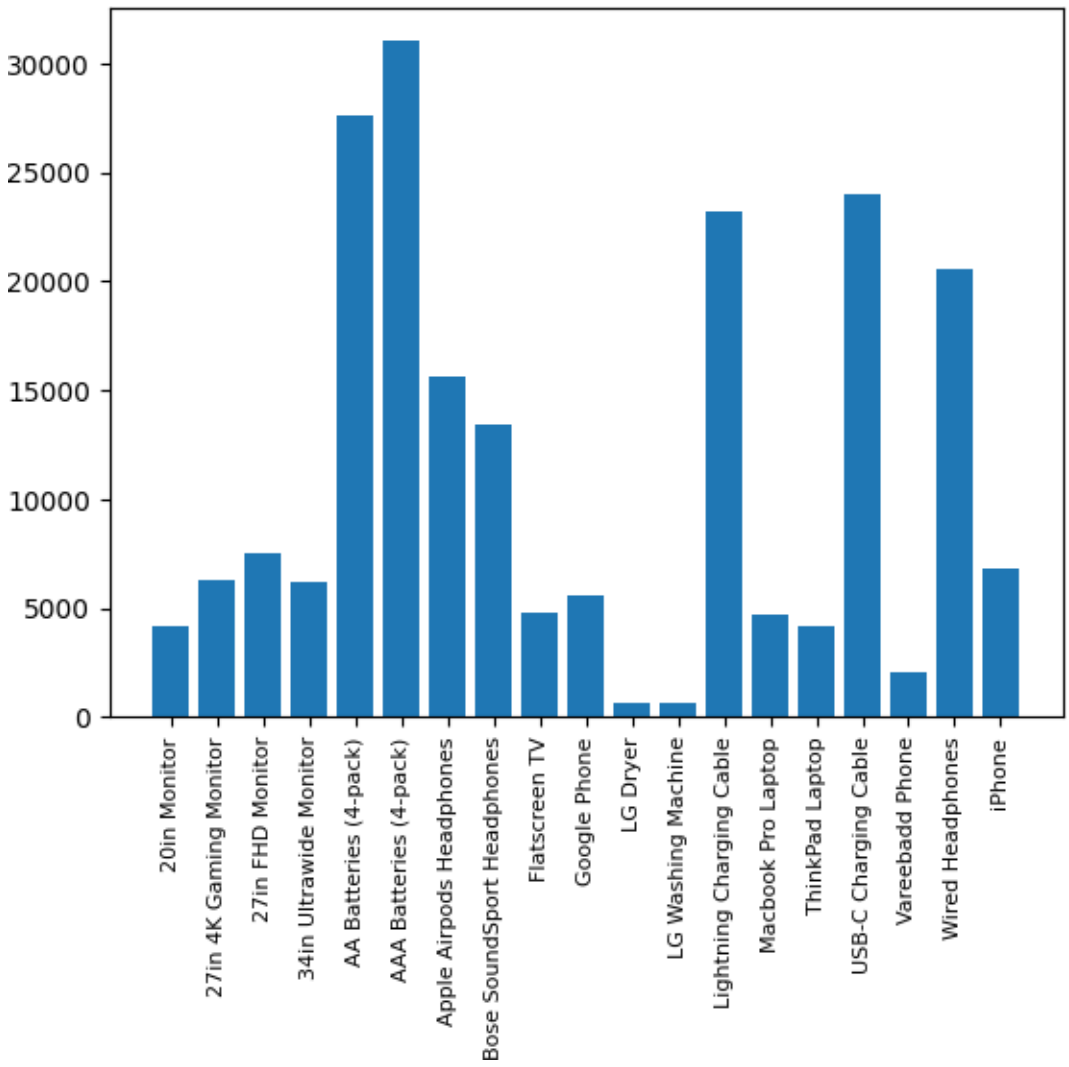
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92

- The most often products sold together are iPhone and Lightning Charging Cable with 1005 transactions.

Question 5: What product sold the most?

[63]:
```
product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']

keys = [pair for pair, df in product_group]
plt.bar(keys, quantity_ordered)
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



- AAA batteries sold the most.

0.7 Case Study 3 :

Create a report for an upcoming board meeting. Go through and analyze the sales data from 2015-2017 in order to generate the requested report. The report should capture the following;

- Revenue by region
- Revenue by sales Rep
- Revenue by products
- Sales trend
- Yearly changes in revenue

[64]:
```
# Import data using pandas
df = pd.read_csv('/content/sales-data.csv')
```

[65]:
```
df.head()
```

[65] :

	Date	SalesRep	Region	Product	Color	Units	Revenue
0	2015-11-06	Julie	East	Sunshine	Blue	4	78.8
1	2015-11-07	Adam	West	Bellen	Clear	4	123.0
2	2015-11-07	Julie	East	Aspen	Clear	1	26.0
3	2015-11-07	Nabil	South	Quad	Clear	2	69.0
4	2015-11-07	Julie	South	Aspen	Blue	2	51.0

[70]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9971 entries, 0 to 9970
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      9971 non-null   object
 1   SalesRep  9971 non-null   object
 2   Region    9971 non-null   object
 3   Product   9971 non-null   object
 4   Color     9971 non-null   object
 5   Units     9971 non-null   int64
 6   Revenue   9971 non-null   float64
dtypes: float64(1), int64(1), object(5)
memory usage: 545.4+ KB
```

[66] :
```
df.describe()
```

[66]:

	Units	Revenue
count	9971.000000	9971.000000
mean	3.388828	91.181513
std	4.320759	120.894473
min	1.000000	21.000000
25%	2.000000	42.900000
50%	2.000000	60.000000
75%	3.000000	76.500000
max	25.000000	1901.750000

There was a total of 9,971 sales entries between 2015-2017 **Units:**

- The minimum number of units sold between 2015-2017 was 1
- The maximum number of units sold between 2015-2017 was 25
- The average number of units sold between 2015-2017 was aproximately 3

Revenue:

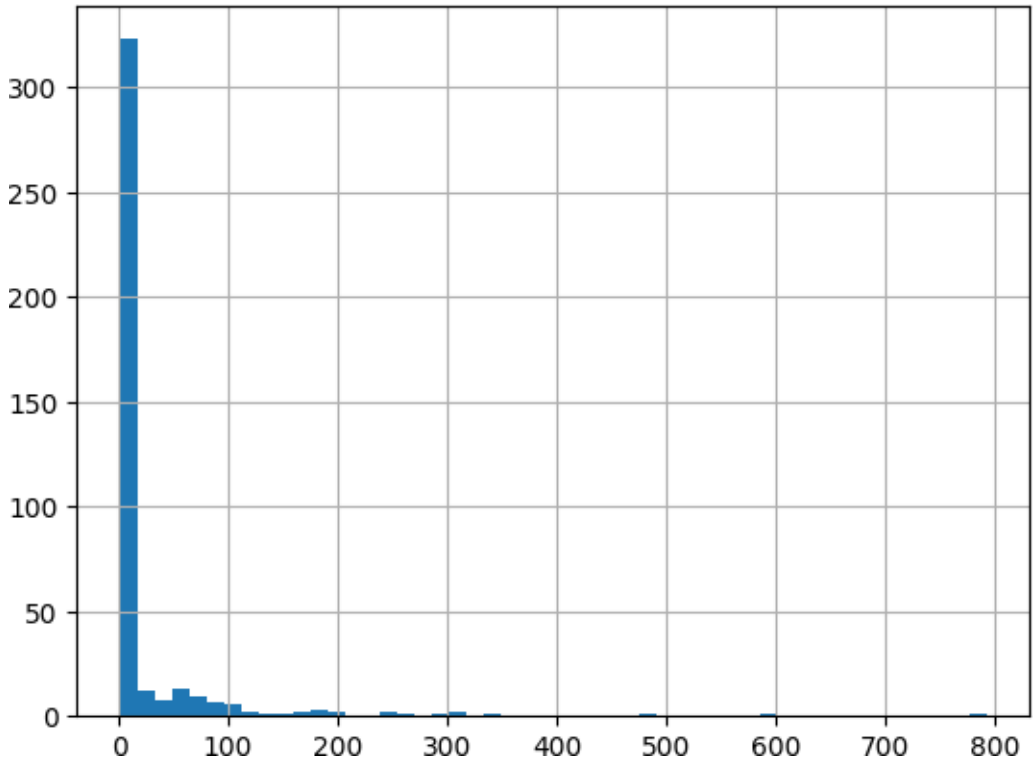
- The least revenue generated between 2015-2017 was 21
- The most revenue between 2015-2017 was approximately 1902

[71]:
```
# Check  missing  entry
df.isna().sum()
```

[71]:
```
Date        0
SalesRep    0
Region      0
Product     0
Color       0
Units       0
Revenue     0
dtype: int64
```

1. Revenue Analysis

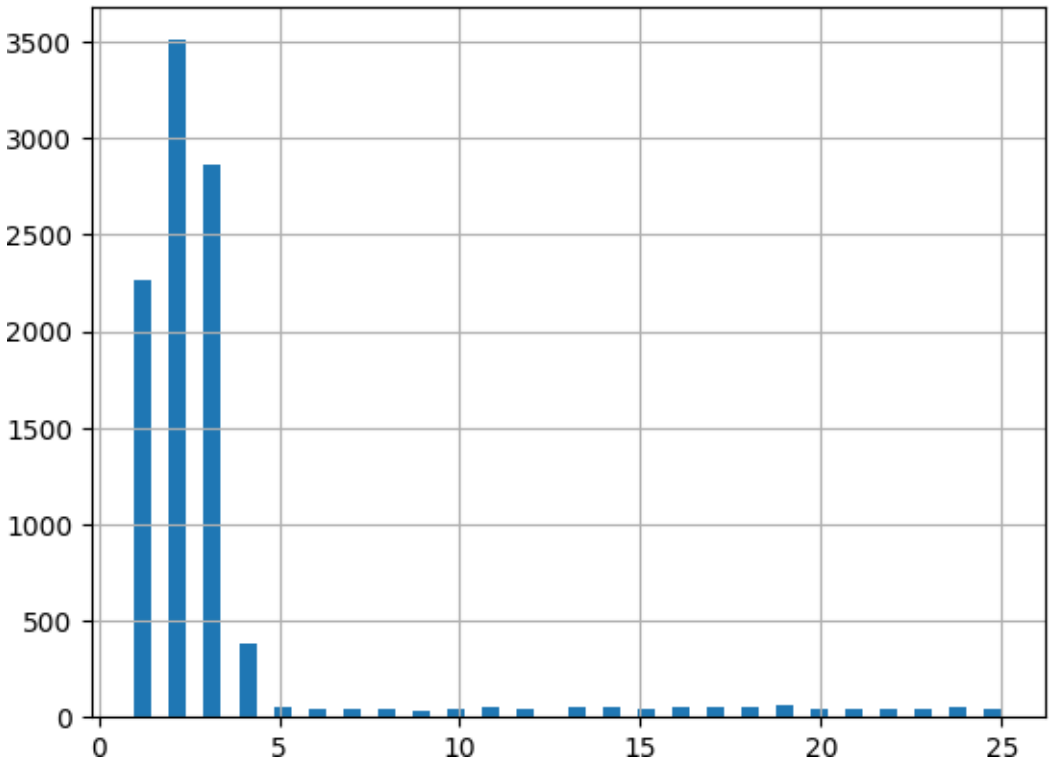
[67] :
```
df['Revenue'].value_counts().hist(bins=50);
```



Most items were sold between 21 - 70.

2. Units Analysis

[68] :
```
df['Units'].hist(bins=50);
```



3. What's the total revenue generated between 2015-2017?

[69] :
```
round(df['Revenue'].sum())
```

[69]: 909171

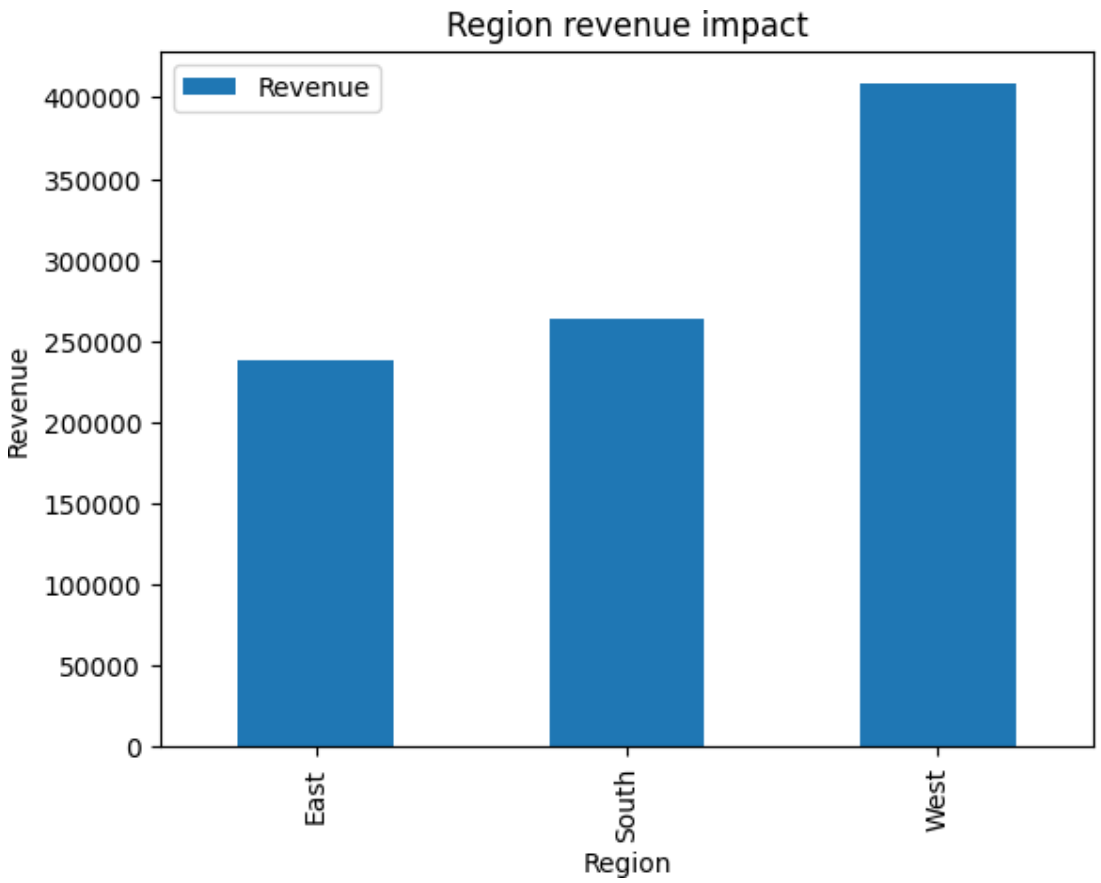
4. Revenue by Region

[72]:
```
region_revenue = pd.DataFrame(df.groupby(by=['Region'])['Revenue'].sum())
region_revenue.sort_values(ascending=False, by='Revenue')
```

[72] :

Region	Revenue
West	408037.58
South	263256.50
East	237876.79

[73] :
```
region_revenue.plot(kind='bar', ylabel='Revenue', title='Region revenue␣
 ↪impact');
```



- West Region generated the most revenue

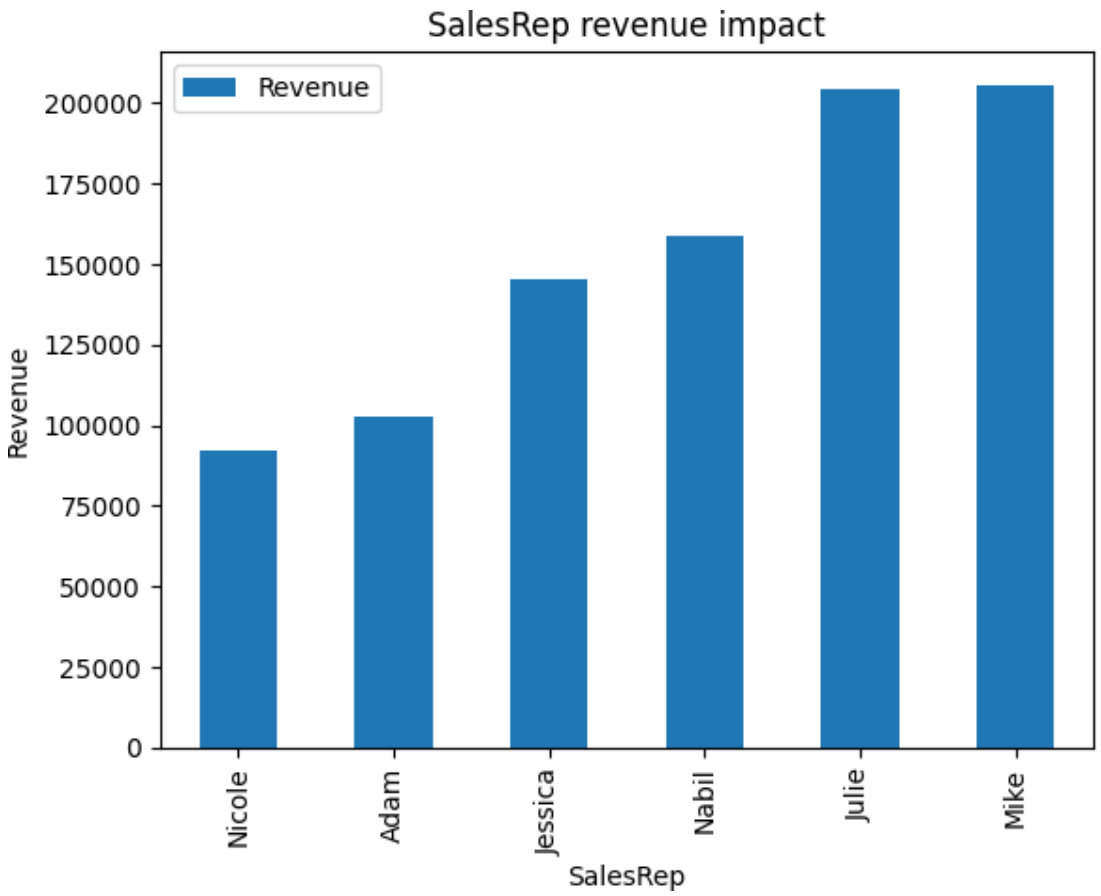
5. Revenue by sales Rep

[74] :
```
sales_rep_revenue = df.groupby(by=['SalesRep'])['Revenue'].sum()
sales_rep_revenue = pd.DataFrame(sales_rep_revenue).sort_values(ascending=True,␣
 ↪by='Revenue')
sales_rep_revenue
```

[74]:

SalesRep	Revenue
Nicole	92026.68
Adam	102715.60
Jessica	145496.28
Nabil	158904.48
Julie	204450.05
Mike	205577.78

[75] :
```
sales_rep_revenue.plot(kind='bar', ylabel='Revenue', title='SalesRep revenue␣
 ↪impact');
```



- Mike Slightly beat Julie in revenue generation

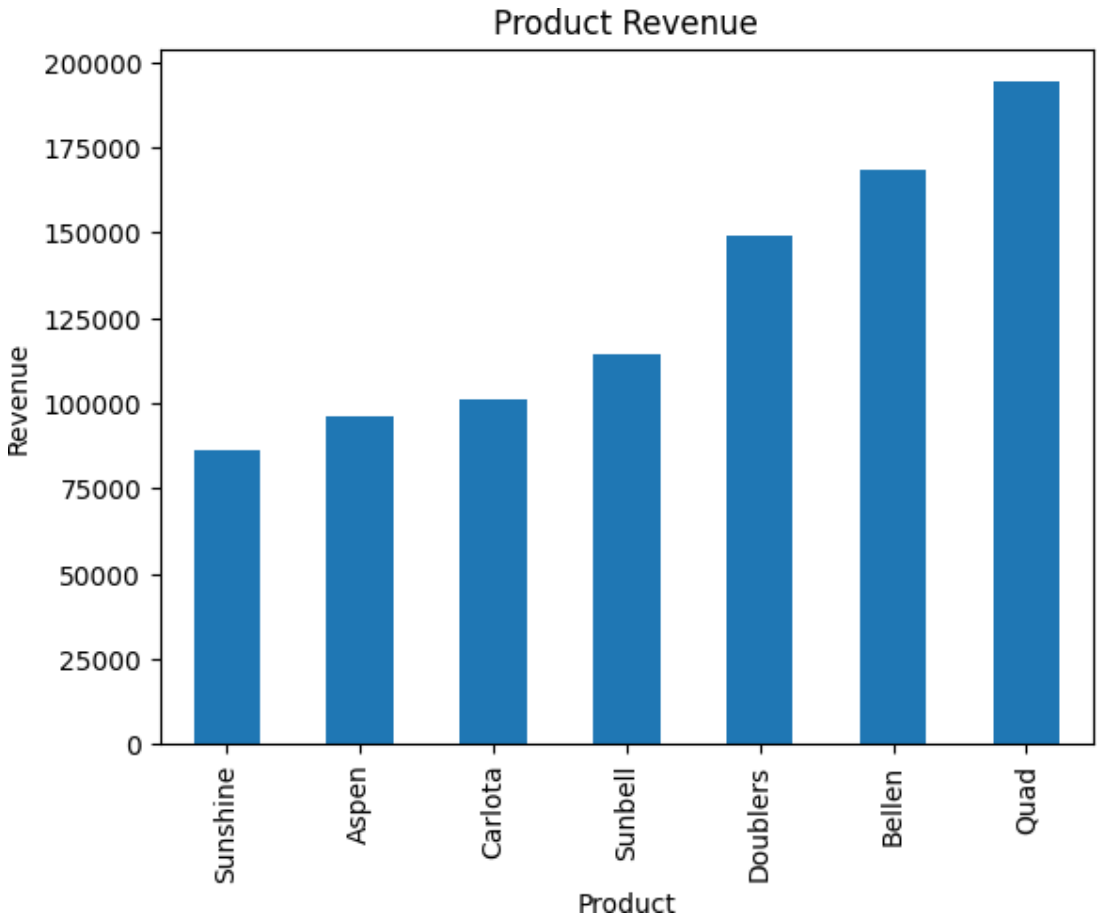
6. Revenue by Products

[76] :
```
product_revenue = df[['Units', 'Revenue','Product']].groupby('Product').sum().
sort_values(ascending=False,by='Units')
product_revenue
```

[76]:

Product	Units	Revenue
Bellen	6579	168175.05
Quad	6223	194032.15
Sunbell	4500	114283.09
Carlota	4371	101272.05
Aspen	4242	96382.80
Sunshine	4229	85983.80
Doublers	3646	149041.93

[77] :
```
product_revenue.groupby(by=['Product'])['Revenue'].sum().
sort_values(ascending=True).plot(
            kind='bar',ylabel='Revenue',title='Product  Revenue');
```



- Quad has highest Revenue.

7. Sales Trend

[78] :
```
# Convert the date column to a datetime object
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day']   = df['Date'].dt.day
```

Plot Yearly Sales Trend

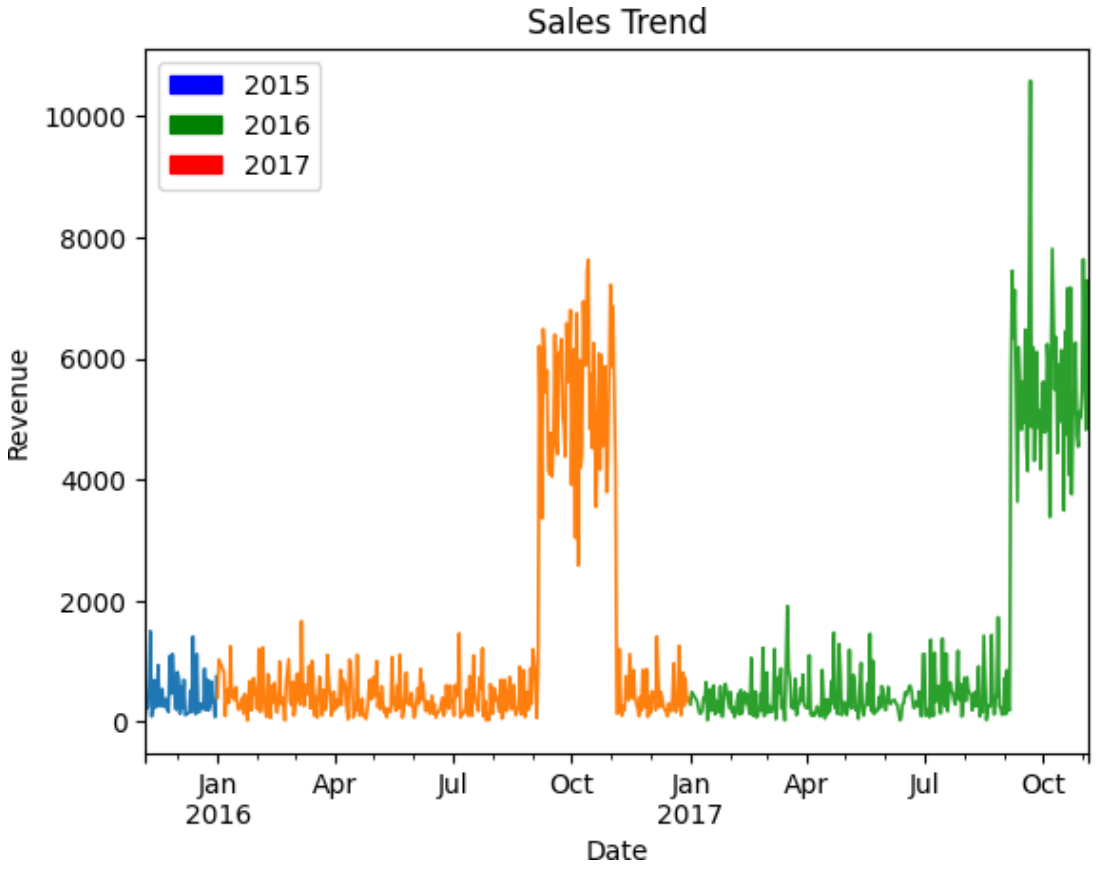
[79] :
```
years = [unique for unique in df.Year.unique()]
years
```

[79]: [2015, 2016, 2017]

[80] :
```
def plot_trend(years:list, df):
    for year in years:
        new_df = df[df['Year'] == year]
        new_df.groupby('Date')['Revenue'].sum().plot(linewidth=1.2,
                                                     ylabel='Revenue',
                                                     xlabel='Date',
                                                     title='Sales Trend');
```

[81] :
```
import matplotlib.patches as patches

year1 = patches.Patch(color='blue', label='2015')
year2 = patches.Patch(color='green', label='2016')
year3 = patches.Patch(color='red', label='2017')
plot_trend(years, df)
plt.legend(handles=[year1,year2,year3], loc=2);
```

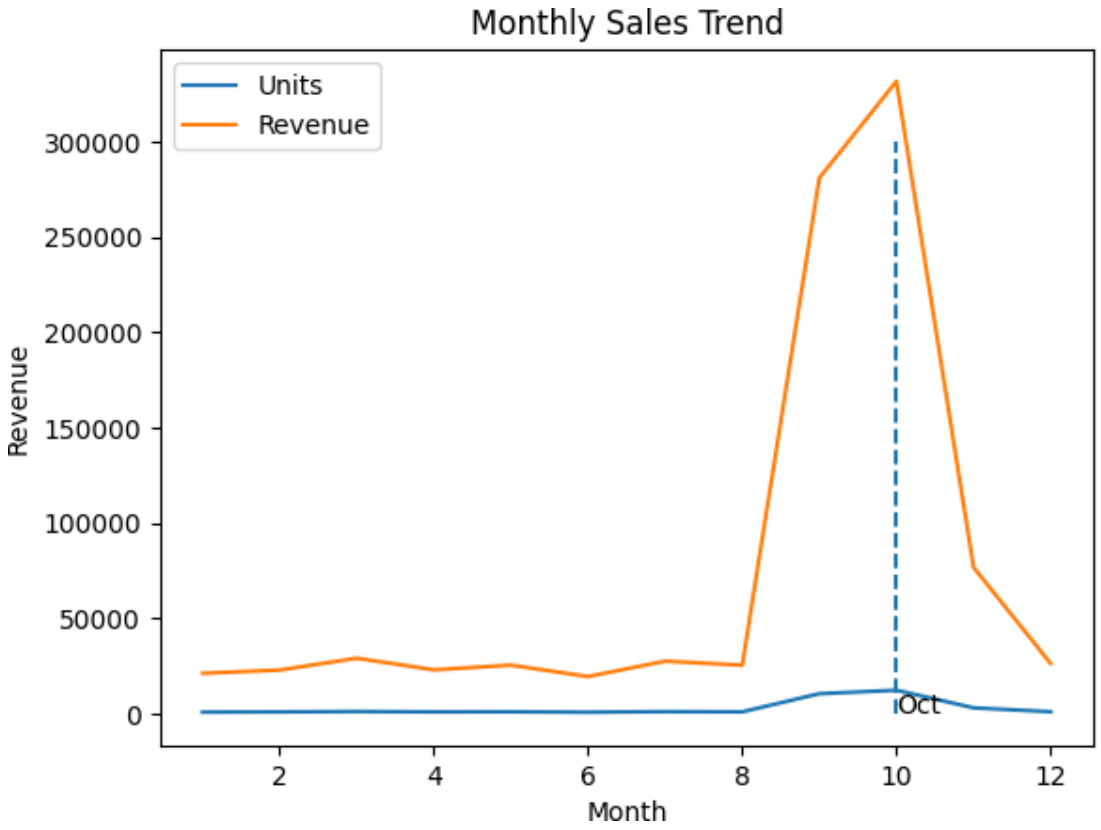


The trend plot looks symmetrical for the months of October in 2017 and 2018 respectively.

Monthly Sales Trend

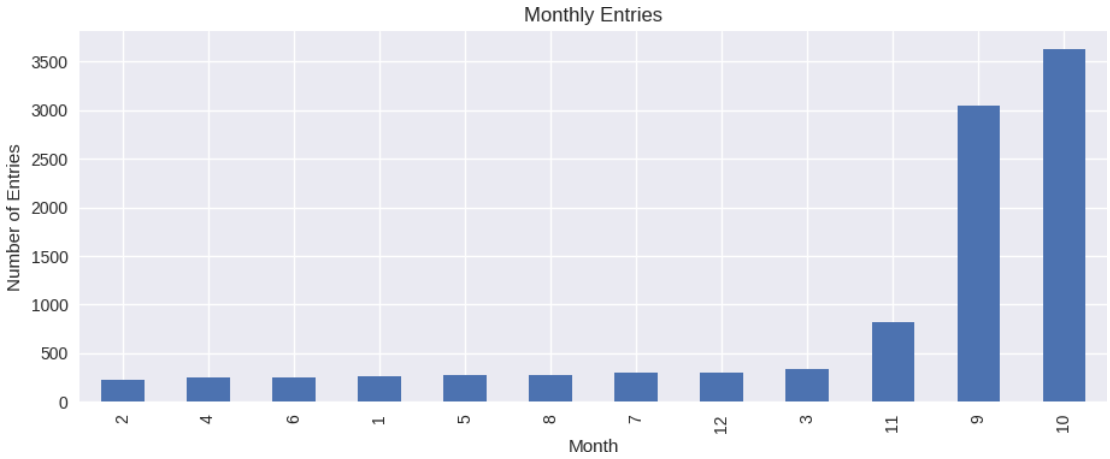
[82] :
```
ax = df[['Month', 'Units', 'Revenue']].groupby('Month').sum().plot(
                                                        title='Monthly␣
 ↪Sales Trend',

 ↪ylabel='Revenue',
                                                        );
ax.vlines(10,1,300000, linestyles='dashed')
ax.annotate('Oct',(10,0));
```



8. How many times was entry made in each month?

[ ]:
```
df['Month'].value_counts().sort_values().plot(kind='bar', xlabel='Month',␣
 ↪ylabel='Number of Entries', title='Monthly Entries');
```



- Highest Entry in October.

9. Monthly Sales

[83] :
```
products = pd.DataFrame(df[['Units','Revenue','Product','Month', 'Region']].
groupby('Month')['Product'].value_counts())
products
```

[83] :

Month	Product	count
1	Bellen	52
	Quad	46
	Sunbell	34
	Sunshine	33
	Aspen	33
...		...
12	Sunbell	43
	Aspen	41
	Sunshine	36
	Carlota	35
	Doublers	32

[84 rows x 1 columns]

[84] :
```
products['No_of_products'] = products['count']

products = products.reset_index()
```

[85] :
```
products.drop('count', inplace=True, axis=1)
products
```

[85]:

	Month	Product	No_of_products
0	1	Bellen	52
1	1	Quad	46
2	1	Sunbell	34
3	1	Sunshine	33
4	1	Aspen	33
..	...	...	...
79	12	Sunbell	43
80	12	Aspen	41
81	12	Sunshine	36
82	12	Carlota	35
83	12	Doublers	32

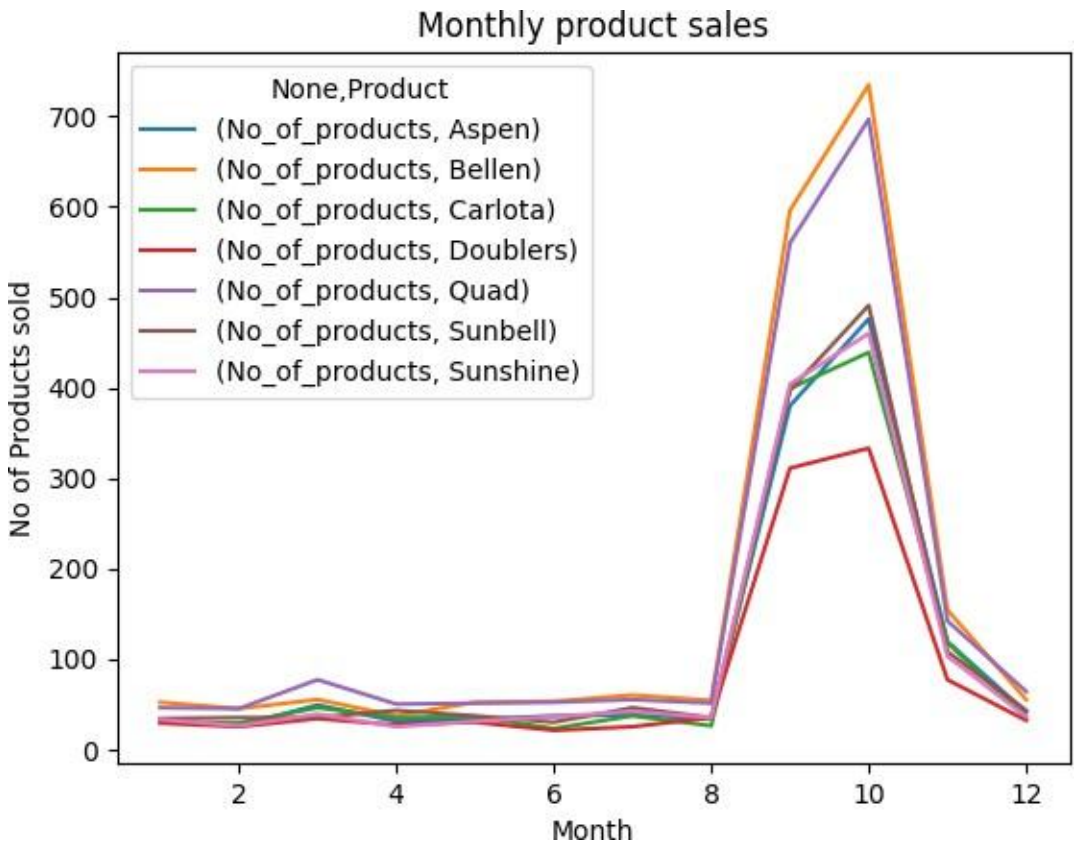
[84 rows x 3 columns]

[87]:
```
products = products.pivot_table(values=['No_of_products'], index=['Month'],
 columns=['Product'], aggfunc= np.sum)
products
```

[87]:

	No_of_products						
Product	Aspen	Bellen	Carlota	Doublers	Quad	Sunbell	Sunshine
Month							
1	33	52	30	29	46	34	33
2	26	45	29	25	45	35	26
3	49	55	46	34	77	35	39
4	31	37	35	27	50	43	25
5	33	52	36	30	51	37	31
6	37	53	23	21	52	30	36
7	38	60	37	25	55	46	42
8	34	54	26	35	51	35	35
9	380	596	399	311	560	397	404
10	476	735	439	333	697	491	460
11	119	154	118	77	142	108	103
12	41	55	35	32	64	43	36

[88]:
```
products.plot(ylabel='No of Products sold', title='Monthly product sales');
```



10. Region Monthly Revenue

[89]:
```
region_sales = pd.DataFrame(df[['Units','Revenue','Product','Month',
 's'Region']]).groupby(['Month','Region'])['Revenue'].sum()
region_sales = pd.DataFrame(region_sales)
region_sales
```

[89]:

Month	Region	Revenue
1	East	5012.34
	South	7551.55
	West	8550.33
2	East	6428.75
	South	5540.10
	West	10864.87
3	East	6082.75
	South	8863.80
	West	14087.99
4	East	6420.63
	South	7647.28

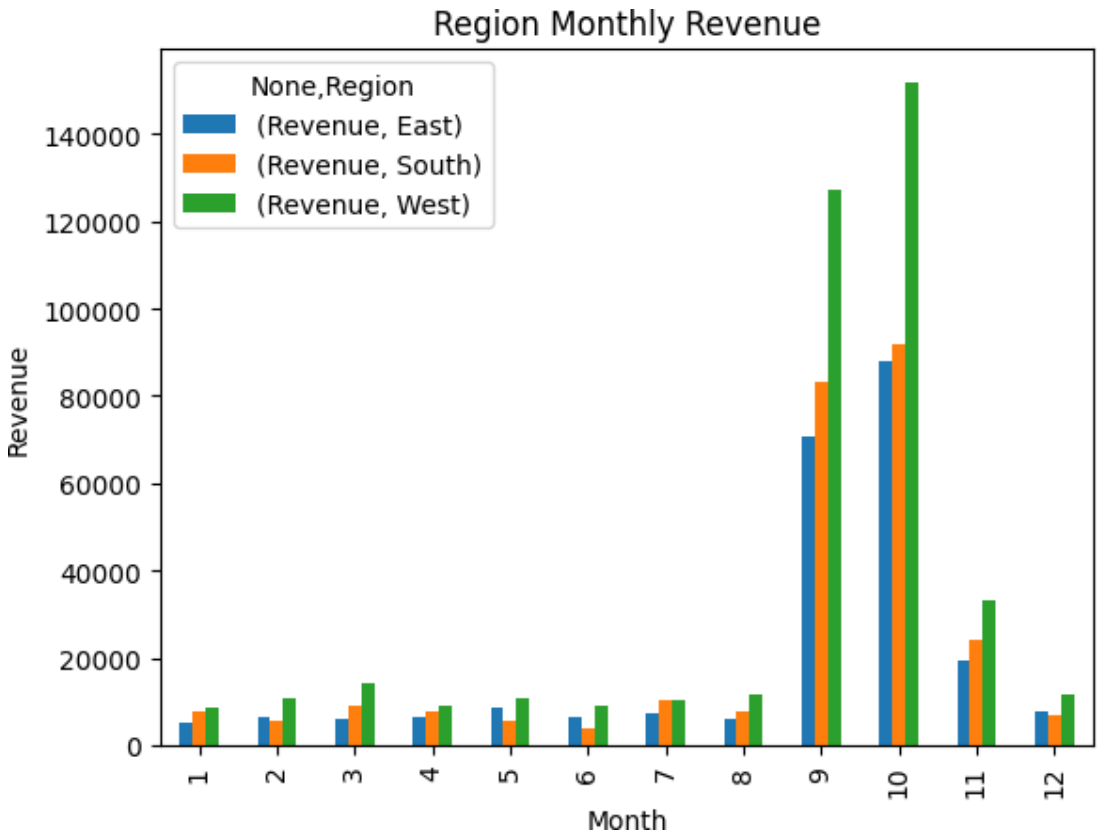
	West	8865.57
5	East	8782.68
	South	5651.30
	West	10962.00
6	East	6442.85
	South	3954.90
	West	9020.65
7	East	7180.45
	South	10155.59
	West	10150.25
8	East	6031.55
	South	7767.60
	West	11567.37
9	East	70532.44
	South	83228.39
	West	127160.06
10	East	87858.60
	South	92034.70
	West	151780.43
11	East	19478.10
	South	24048.59
	West	33196.52
12	East	7625.65
	South	6812.70
	West	11831.54

[90]:
```
region_sales = region_sales.reset_index()
region_sales = region_sales.pivot_table(values=['Revenue'], index=['Month'],
 columns=['Region'], aggfunc= np.sum)
region_sales
```

[90]:

	Revenue		
Region	East	South	West
Month			
1	5012.34	7551.55	8550.33
2	6428.75	5540.10	10864.87
3	6082.75	8863.80	14087.99
4	6420.63	7647.28	8865.57
5	8782.68	5651.30	10962.00
6	6442.85	3954.90	9020.65
7	7180.45	10155.59	10150.25
8	6031.55	7767.60	11567.37
9	70532.44	83228.39	127160.06
10	87858.60	92034.70	151780.43
11	19478.10	24048.59	33196.52
12	7625.65	6812.70	11831.54

[91]:
```
region_sales.plot(kind='bar', ylabel='Revenue', title='Region Monthly Revenue');
```



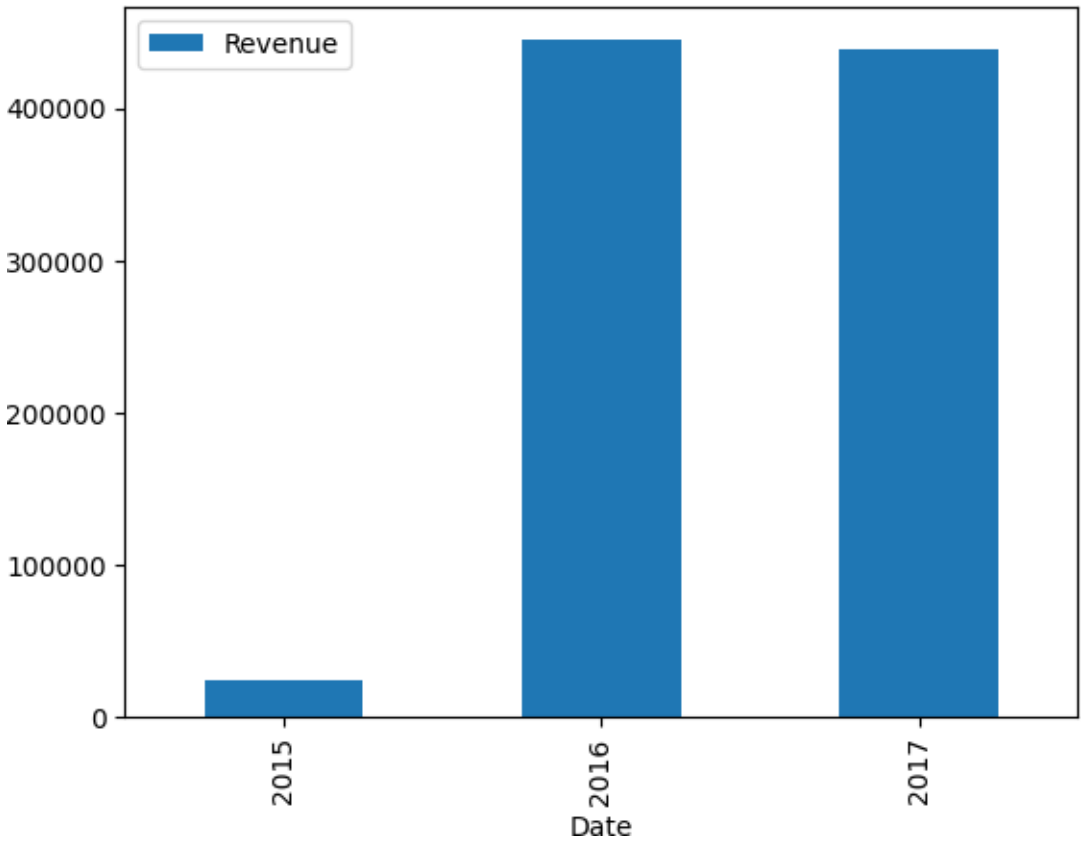
11. Yearly changes in revenue

[92]:
```
changes = pd.DataFrame(df.groupby([df.Date.dt.year])['Revenue'].sum())
changes
```

[92] :

Date	Revenue
2015	24883.84
2016	444701.72
2017	439585.31

[93] :
```
changes.sort_values('Date').plot(kind='bar');
```



12. Top 3 products

[94] :
```
product_revenue
```

[94]:

Product	Units	Revenue
Bellen	6579	168175.05
Quad	6223	194032.15
Sunbell	4500	114283.09
Carlota	4371	101272.05
Aspen	4242	96382.80
Sunshine	4229	85983.80
Doublers	3646	149041.93

Bellen, Quad and Sunbell

13. The most productive sales Rep in the respective years.

[95] :
```
salesReps = df[['SalesRep','Year','Revenue','Units']]
salesReps = pd.DataFrame(salesReps.groupby(['Year','SalesRep'])['Revenue'].
sum())
```

```
salesReps.sort_values(by=['Year','Revenue'],  ascending=False)
```

[95]:

Year	SalesRep	Revenue
2017	Julie	99727.32
	Mike	96062.19
	Nabil	81079.23
	Jessica	69479.74
	Adam	49712.19
	Nicole	43524.64
2016	Mike	104590.64
	Julie	98895.58
	Nabil	74576.22
	Jessica	71469.42
	Adam	49184.21
	Nicole	45985.65
2015	Julie	5827.15
	Mike	4924.95
	Jessica	4547.12
	Adam	3819.20
	Nabil	3249.03
	Nicole	2516.39

Recommendation:

- The best months for sales are September, October and November.
- The company should look into creating jingles during these periods to further maximize profit.
- Focus the ad targeted audience on East and South Regions.
- Bellen and Quad sell most during these periods consider getting more of them.