$ Exploratory Data Analysis (EDA) on Sales_Pipeline By Dolly Singh$

SALES_PIPELINE DATASET

Few lines describe about sales pipeline

#1.A sales pipeline is an organized, visual way of tracking potential buyers as they progress through different stages in the

#purchasing process and buyer's journey.

#2.pipelines are visualized as a horizontal bar (sometimes as a funnel) divided into the various stages of a company's sales process.

#3.Leads and prospects are moved from one stage to the next as they maneuver through the sales process.

#4.e.g. when reps receive a response to outreach like a cold email or when a potential customer is marked as a qualified or unqualified lead.

#6 stage of sales pipeline:-

#1.generate new lead.

#2.Qualifying prospects.

#3.Engaging with a lead.

#4.Building a relationship.

#5.Negotiating with the lead.

#6.Closing the deal -->

Import usefull python library

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.io as pio
sns.set_theme(style="darkgrid")

link="C:\\Users\\rites\\Downloads\\sales_pipeline.csv"
sp= pd.read_csv(link)
```

checking the top 10 columns of the dataset

```
sp.head()
```

```
  opportunity_id      sales_agent         product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht          GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  starting_date  close_date  close_value
0    2016-10-20  2017-03-01       1054.0
1    2016-10-25  2017-03-11       4514.0
2    2016-10-25  2017-03-07         50.0
3    2016-10-25  2017-03-09        588.0
4    2016-10-25  2017-03-02        517.0
```

checking the bottom 10 columns of the dataset

```
sp.tail()
```

```
     opportunity_id        sales_agent      product account
deal_stage  \
8795       9MIWFW5J  Versie Hillebrand  MG Advanced     NaN
Prospecting
8796       6SLKZ8FI  Versie Hillebrand  MG Advanced     NaN
Prospecting
8797       LIB4KUZJ  Versie Hillebrand  MG Advanced     NaN
Prospecting
8798       18IUIUK0  Versie Hillebrand  MG Advanced     NaN
Prospecting
8799       8I5ONXJX  Versie Hillebrand  MG Advanced     NaN
Prospecting

     engage_date close_date  close_value
8795         NaN        NaN          NaN
8796         NaN        NaN          NaN
8797         NaN        NaN          NaN
8798         NaN        NaN          NaN
8799         NaN        NaN          NaN
```

checking the shape of the dataset

```
sp.shape
```

```
(8800, 8)
```

checking the column name of the dataset

```
sp.columns
```

```
Index(['opportunity_id', 'sales_agent', 'product', 'account',
'deal_stage',
       'engage_date', 'close_date', 'close_value'],
      dtype='object')
```

show the type dataset

```
print(type(sp))
```

```
<class 'pandas.core.frame.DataFrame'>
```

show the null column

```
sp.isna().sum()
```

```
opportunity_id       0
sales_agent          0
product              0
account           1425
deal_stage           0
engage_date        500
close_date        2089
close_value       2089
dtype: int64
```

show information of the datset

```
sp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8800 entries, 0 to 8799
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------           --------------  -----
 0   opportunity_id  8800 non-null   object
 1   sales_agent     8800 non-null   object
 2   product         8800 non-null   object
 3   account         7375 non-null   object
 4   deal_stage      8800 non-null   object
 5   engage_date     8300 non-null   object
 6   close_date      6711 non-null   object
 7   close_value     6711 non-null   float64
dtypes: float64(1), object(7)
memory usage: 550.1+ KB
```

check statistical function

```
sp.describe()
```

```
        close_value
count   6711.000000
mean    1490.915512
std     2320.670773
min        0.000000
25%        0.000000
50%      472.000000
75%     3225.000000
max    30288.000000
```

----------DATA CLEANING----------

```
sp.isna().sum()
```

```
opportunity_id       0
sales_agent          0
product              0
account           1425
deal_stage           0
engage_date        500
close_date        2089
close_value       2089
dtype: int64
```

check the percentage of null value column

```
x=sp.isna().sum()
x=dict(x)
for i,j in x.items():
    print(i,round(j/8800*100,2))
```

```
opportunity_id 0.0
sales_agent 0.0
product 0.0
account 16.19
deal_stage 0.0
engage_date 5.68
close_date 23.74
close_value 23.74
```

```
sp.head()
```

```
  opportunity_id      sales_agent          product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht          GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  engage_date  close_date  close_value
0  2016-10-20  2017-03-01       1054.0
1  2016-10-25  2017-03-11       4514.0
2  2016-10-25  2017-03-07         50.0
3  2016-10-25  2017-03-09        588.0
4  2016-10-25  2017-03-02        517.0
```

fill the account column

```
# data type of account column
sp["account"].dtype
```

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

```
x=sp["account"].mode()
x
```

```
0    Hottechi
Name: account, dtype: object

sp["account"].fillna(x[0],inplace=True)

sp["account"].isna().sum()

0

sp.head()

  opportunity_id      sales_agent         product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht          GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  engage_date  close_date  close_value
0  2016-10-20  2017-03-01       1054.0
1  2016-10-25  2017-03-11       4514.0
2  2016-10-25  2017-03-07         50.0
3  2016-10-25  2017-03-09        588.0
4  2016-10-25  2017-03-02        517.0

sp.tail()

     opportunity_id        sales_agent      product   account
deal_stage  \
8795       9MIWFW5J  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8796       6SLKZ8FI  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8797       LIB4KUZJ  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8798       18IUIUK0  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8799       8I5ONXJX  Versie Hillebrand  MG Advanced  Hottechi
Prospecting

     engage_date close_date  close_value
8795         NaN        NaN          NaN
8796         NaN        NaN          NaN
8797         NaN        NaN          NaN
```

```
8798          NaN         NaN           NaN
8799          NaN         NaN           NaN
```

fill the engage_date column

```
sp["engage_date"].dtype
```

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

```
x =sp["engage_date"].mode()
x
```

```
0    2017-07-22
Name: engage_date, dtype: object
```

```
sp["engage_date"].fillna(x[0],inplace=True)
```

```
sp["engage_date"].isna().sum()
```

```
0
```

```
sp.head()
```

```
  opportunity_id      sales_agent          product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht           GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  engage_date  close_date  close_value
0  2016-10-20  2017-03-01       1054.0
1  2016-10-25  2017-03-11       4514.0
2  2016-10-25  2017-03-07         50.0
3  2016-10-25  2017-03-09        588.0
4  2016-10-25  2017-03-02        517.0
```

```
sp.tail()
```

```
     opportunity_id        sales_agent      product   account
deal_stage  \
8795       9MIWFW5J  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8796       6SLKZ8FI  Versie Hillebrand  MG Advanced  Hottechi
```

```
Prospecting
8797       LIB4KUZJ  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8798       18IUIUK0  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8799       8I5ONXJX  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
```

```
     engage_date close_date  close_value
8795  2017-07-22        NaN          NaN
8796  2017-07-22        NaN          NaN
8797  2017-07-22        NaN          NaN
8798  2017-07-22        NaN          NaN
8799  2017-07-22        NaN          NaN
```

fill the close_date column

```
sp["close_date"].dtype
```

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

```
x=sp["close_date"].mode()
x
```

```
0    2017-05-22
Name: close_date, dtype: object
```

```
sp["close_date"].fillna(x[0],inplace=True)
```

```
sp["close_date"].isna().sum()
```

```
0
```

```
sp.head()
```

```
  opportunity_id      sales_agent         product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht          GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  engage_date  close_date  close_value
0  2016-10-20  2017-03-01       1054.0
```

```
1  2016-10-25  2017-03-11       4514.0
2  2016-10-25  2017-03-07         50.0
3  2016-10-25  2017-03-09        588.0
4  2016-10-25  2017-03-02        517.0
```

```
sp.tail()
```

```
     opportunity_id        sales_agent      product   account
deal_stage  \
8795       9MIWFW5J  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8796       6SLKZ8FI  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8797       LIB4KUZJ  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8798       18IUIUK0  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8799       8I5ONXJX  Versie Hillebrand  MG Advanced  Hottechi
Prospecting

     engage_date  close_date  close_value
8795  2017-07-22  2017-05-22          NaN
8796  2017-07-22  2017-05-22          NaN
8797  2017-07-22  2017-05-22          NaN
8798  2017-07-22  2017-05-22          NaN
8799  2017-07-22  2017-05-22          NaN
```

fill the data of close value column

```
sp["close_value"].dtype
```

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

```
M=sp["close_value"].mean()
M
```

```
1490.9155118462227
```

```
sp["close_value"].median()
```

```
472.0
```

```
sp["close_value"].fillna(M,inplace=True)
```

```
sp["close_value"].isna().sum()
```

```
0
```

```
sp.head()
```

```
  opportunity_id      sales_agent         product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht          GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  engage_date  close_date  close_value
0  2016-10-20  2017-03-01       1054.0
1  2016-10-25  2017-03-11       4514.0
2  2016-10-25  2017-03-07         50.0
3  2016-10-25  2017-03-09        588.0
4  2016-10-25  2017-03-02        517.0

sp.tail()

     opportunity_id        sales_agent      product   account
deal_stage  \
8795       9MIWFW5J  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8796       6SLKZ8FI  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8797       LIB4KUZJ  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8798       18IUIUK0  Versie Hillebrand  MG Advanced  Hottechi
Prospecting
8799       8I5ONXJX  Versie Hillebrand  MG Advanced  Hottechi
Prospecting

     engage_date  close_date  close_value
8795  2017-07-22  2017-05-22  1490.915512
8796  2017-07-22  2017-05-22  1490.915512
8797  2017-07-22  2017-05-22  1490.915512
8798  2017-07-22  2017-05-22  1490.915512
8799  2017-07-22  2017-05-22  1490.915512
```

==========DONE==========

----------DATA MANIPULATION----------

change the engage_date column name to starting_data

```
x=sp.rename(columns={"engage_date":"starting_date"},inplace=True)
x

sp.head()
```

```
  opportunity_id      sales_agent         product  account deal_stage
\
0       1C1I7A6R      Moses Frase  GTX Plus Basic  Cancity        Won

1       Z063OYW0  Darcel Schlecht          GTXPro    Isdom        Won

2       EC4QE1BX  Darcel Schlecht      MG Special  Cancity        Won

3       MV1LWRNH      Moses Frase       GTX Basic  Codehow        Won

4       PE84CX4O        Zane Levy       GTX Basic   Hatfan        Won


  starting_date  close_date  close_value
0    2016-10-20  2017-03-01       1054.0
1    2016-10-25  2017-03-11       4514.0
2    2016-10-25  2017-03-07         50.0
3    2016-10-25  2017-03-09        588.0
4    2016-10-25  2017-03-02        517.0
```

1. we are finding the sales_agent to close_value on sales_pipeline

```
# check the unique name of sales_agent
sp["sales_agent"].unique()
```

```
array(['Moses Frase', 'Darcel Schlecht', 'Zane Levy', 'Anna Snelling',
       'Vicki Laflamme', 'Markita Hansen', 'Niesha Huffines',
       'James Ascencio', 'Gladys Colclough', 'Maureen Marcano',
       'Hayden Neloms', 'Rosalina Dieter', 'Versie Hillebrand',
       'Daniell Hammack', 'Elease Gluck', 'Violet Mclelland',
       'Kami Bicknell', 'Rosie Papadopoulos', 'Kary Hendrixson',
       'Reed Clapper', 'Wilburn Farren', 'Garret Kinder',
       'Marty Freudenburg', 'Cassey Cress', 'Lajuana Vencill',
       'Boris Faz', 'Donn Cantrell', 'Corliss Cosme', 'Cecily
Lampkin',
       'Jonathan Berthelot'], dtype=object)
```

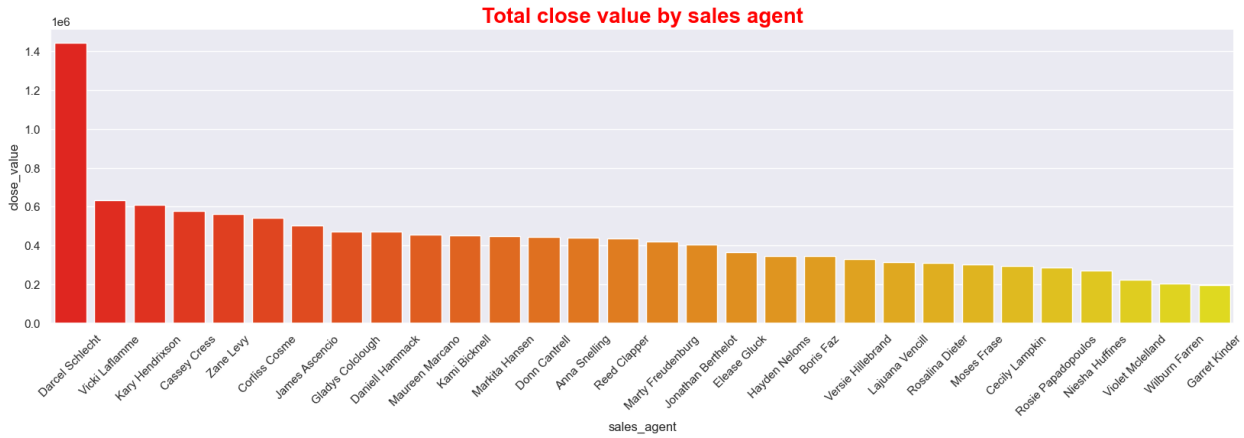
```
# 1.1 find the total close_value by sales_agent and create a bar graph
x=sp.groupby("sales_agent")["close_value"].sum().reset_index()
x =x.sort_values("close_value",ascending=False)
x.head(10)
```

```
         sales_agent   close_value
6    Darcel Schlecht  1.442452e+06
26    Vicki Laflamme  6.334512e+05
15   Kary Hendrixson  6.078623e+05
2       Cassey Cress  5.772168e+05
29         Zane Levy  5.612686e+05
4       Corliss Cosme  5.418002e+05
12     James Ascencio  5.044788e+05
10  Gladys Colclough  4.724018e+05
5     Daniell Hammack  4.715749e+05
19    Maureen Marcano  4.577409e+05
```

-Data Visualzation----------

```
plt.figure(figsize=(20,5))
sns.barplot(data=x,x="sales_agent",y="close_value",palette="autumn")
plt.xticks(rotation=45)
plt.title("Total close value by sales
```

```
agent",fontweight="bold",fontsize=20,color="red")
plt.show()
```



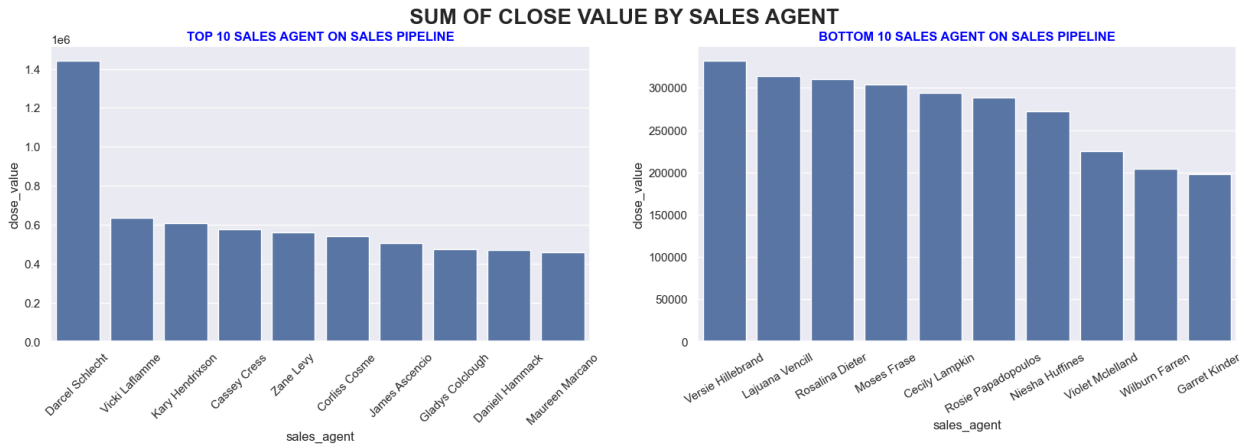
insights

#Top Performers:# David Schwab: Highest close value, significantly above 1.4.# Katy Jessica: Following closely with a strong performance#. Kevin Williamson: Also performing well, with a value near 1.#2. Middle Ran#ge: Agents like Cameron Cruz and Zane Levy show moderate performance, with values around# 0.8 to #1.0. Lower Perfor#mers: The lower end of the scale displays agents with values below 0.5. Notable names i#nclude Cady Lampley and Ronald Marshall, indicating lower performance in closing sales.

#1.2find the top 10 sales agent and bottom 10 sales agent by close value on sales pipeline?

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.barplot(data=x.head(10),x="sales_agent",y="close_value")
plt.title("TOP 10 SALES AGENT ON SALES
PIPELINE",fontweight="bold",color="blue")
plt.xticks(rotation=45)
plt.subplot(1,2,2)
sns.barplot(data=x.tail(10),x="sales_agent",y="close_value")
plt.title("BOTTOM 10 SALES AGENT ON SALES
PIPELINE",fontweight="bold",color="blue")
plt.suptitle("SUM OF CLOSE VALUE BY SALES
AGENT",fontweight="bold",fontsize=20)
plt.xticks(rotation=30)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



insights

#Top 10 and Bottom 10 Sales Agents:

Two bar plots are created:# Top 10 Sales Agents on Sales Pipeline: Displays the sales agents with the highest close value#. Bottom 10 Sales Agents on Sales Pipeline: Displays the sales agents with the lowest close value.

2.Find out the product to close_value on sales pipeline

```
# show the total number of close value by product
x=sp.groupby("product")["close_value"].sum().reset_index()
x
```

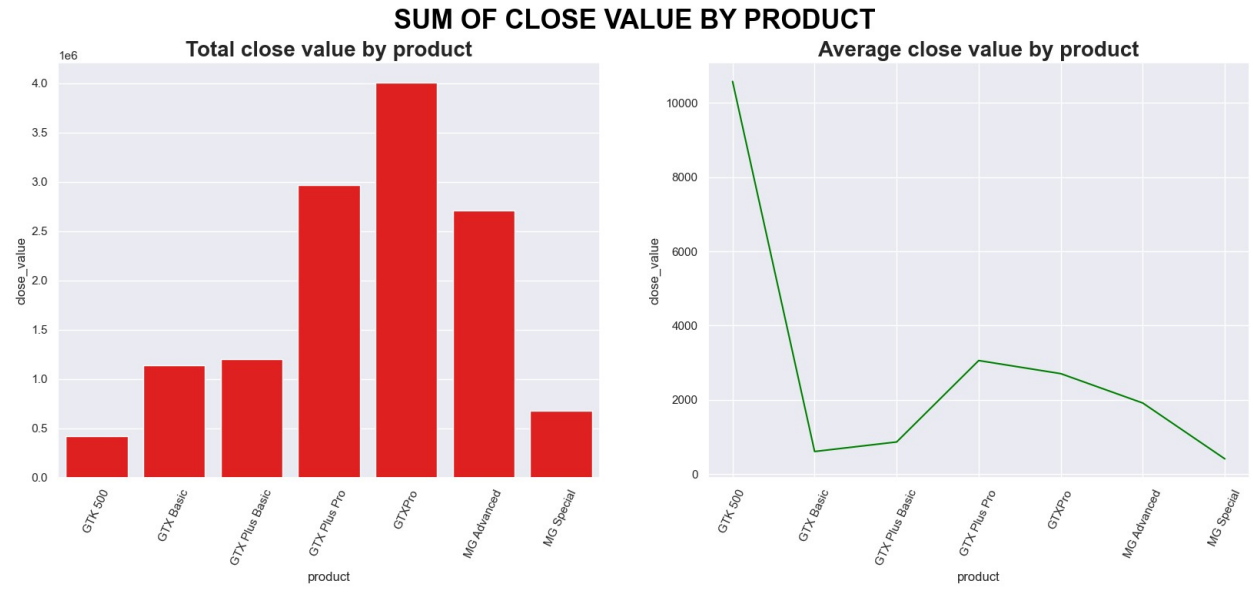
	product	close_value
0	GTK 500	4.229757e+05
1	GTX Basic	1.140357e+06
2	GTX Plus Basic	1.200259e+06
3	GTX Plus Pro	2.962125e+06
4	GTXPro	4.007053e+06
5	MG Advanced	2.705407e+06
6	MG Special	6.818798e+05

```
# show the average number of close value by product
y=sp.groupby("product")["close_value"].mean().reset_index()
y
```

```
          product   close_value
0         GTK 500  10574.393317
1       GTX Basic    611.123617
2  GTX Plus Basic    867.866197
3    GTX Plus Pro   3060.046652
4          GTXPro   2707.468152
5     MG Advanced   1916.010827
6      MG Special    413.010199
```

```
plt.figure(figsize=(20,7))
plt.subplot(1,2,1)
sns.barplot(data=x,x="product",y="close_value",color="red")
plt.title("Total close value by
product",fontweight="bold",fontsize=20)
plt.xticks(rotation=65)
plt.subplot(1,2,2)
sns.lineplot(data=y,x="product",y="close_value",color="green")
plt.title("Average close value by
product",fontweight="bold",fontsize=20)
plt.xticks(rotation=65)
plt.suptitle("SUM OF CLOSE VALUE BY
PRODUCT",fontweight="bold",fontsize=25,color="black")
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



insights

Graph 1: Total Close Value by Product

Type: Bar Chart X-axis: Different Products GTX 1650 GTX 1660 GTX 1660 Super GTX 2060 GTX 2070 Super RTX 3060 Y-axis: Total Close Value Key Observations: The highest close value is for the GTX 2070 Super. The GTX 1650 has the lowest total close value. Graph 2: Average Close Value by Product

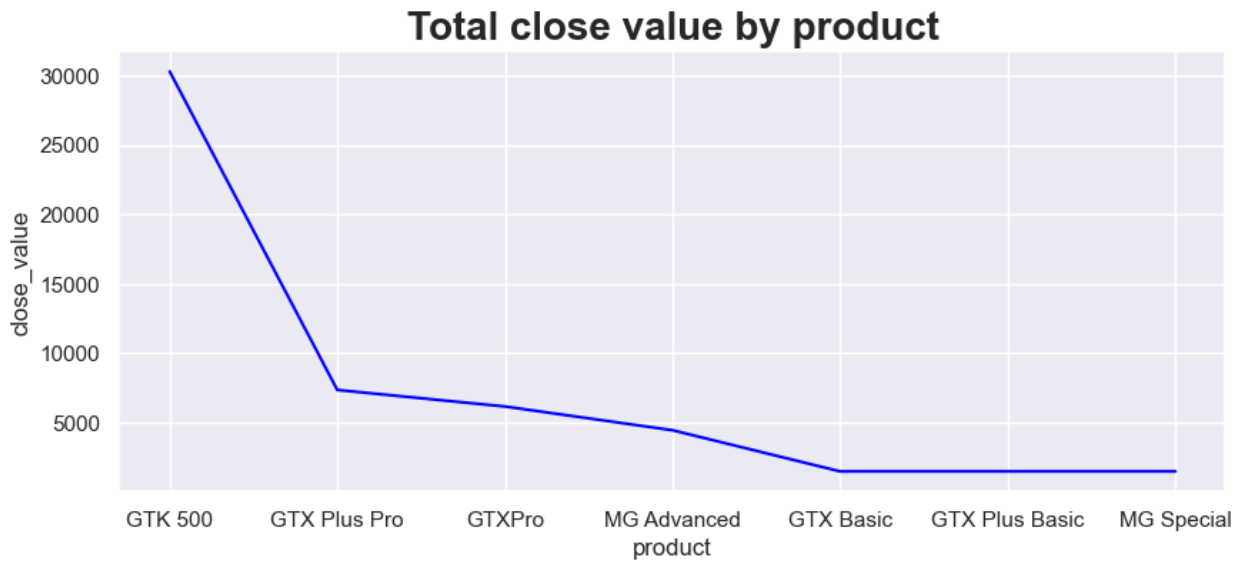
Type: Line Chart X-axis: Different Products (same as above) Y-axis: Average Close Value Key Observations: There is a noticeable drop in average close value from GTX 1650 to GTX 1660. The average value for GTX 2070 Super appears to be stable and higher compared to the other products.

```
#crate a pie chart and show the maximum close value by product?
a =sp.groupby("product")["close_value"].max().reset_index()
a=a.sort_values("close_value",ascending=False)
a
```

	product	close_value
0	GTK 500	30288.000000
3	GTX Plus Pro	7356.000000
4	GTXPro	6166.000000
5	MG Advanced	4456.000000
1	GTX Basic	1490.915512
2	GTX Plus Basic	1490.915512
6	MG Special	1490.915512

```
plt.figure(figsize=(10,4))
sns.lineplot(data=a,x="product",y="close_value",color="blue")
plt.title("Total close value by
product",fontweight="bold",fontsize=20)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



insights

GTK 500: Highest total close value (~30,000). GTK Plus Pro: A sharp decline in close value compared to GTK 500. GTXPro: Continues the downward trend, though less steep than GTK Plus Pro. MG Advanced Product: Maintains a mid-range close value. GTX Basic & GTX Plus Basic: Much lower close values, showing limited market impact. MG Special: Lowest total close value, indicating minimal engagement or sales. Trends The graph shows a steep decline in total close values with only the first few products having substantial values. After the GTK 500, products tend to have a diminished market presence.

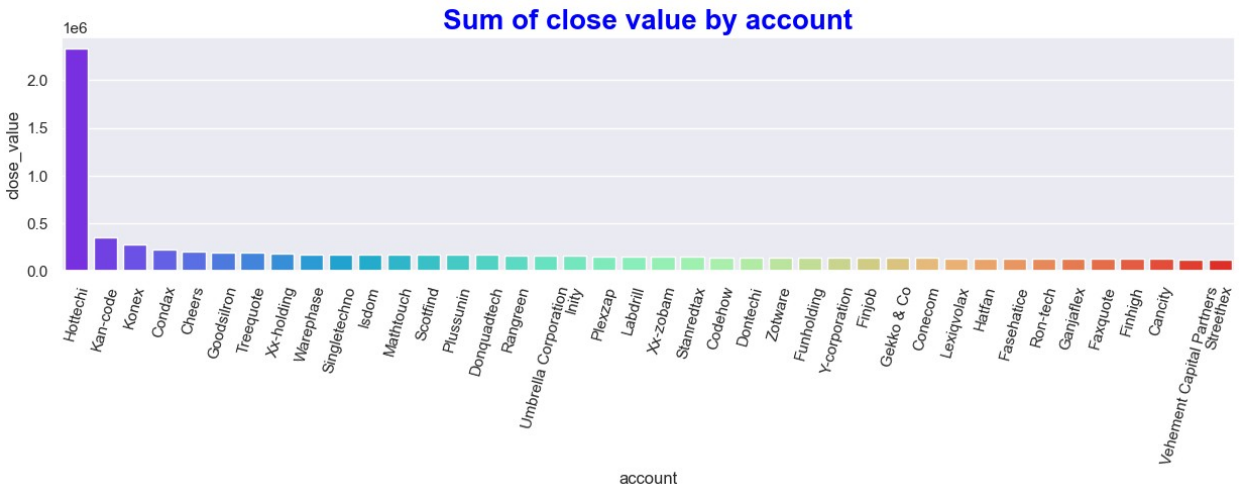
3.Find out the Account to close_value on sales pipeline

```
# find the total account by close value on sales pipeline?
x=sp.groupby("account")["close_value"].sum().reset_index()
x=x.sort_values("close_value",ascending=False)
x
```

```
          account   close_value
34       Hottechi  2.329948e+06
40       Kan-code  3.548732e+05
42          Konex  2.796814e+05
11         Condax  2.228101e+05
9          Cheers  2.099473e+05
..            ...           ...
67         Sumace  7.630507e+04
6   Bluth Company  7.330307e+04
79     Zathunicon  6.605241e+04
18        Donware  6.110975e+04
29        Golddex  6.057749e+04

[85 rows x 2 columns]
```

```
plt.figure(figsize=(15,3))
sns.barplot(data=x.head(40),x="account",y="close_value",palette="rainbow")
plt.title("Sum of close value by account",fontweight="bold",fontsize=20,color="blue")
plt.xticks(rotation=75)
plt.show()
```




```
# find the top 5 account  and bottom 5 bottom by close value on sales
pipeline?
x=sp.groupby("account")["close_value"].sum().reset_index()
x=x.sort_values("close_value",ascending=True).head()
x
```

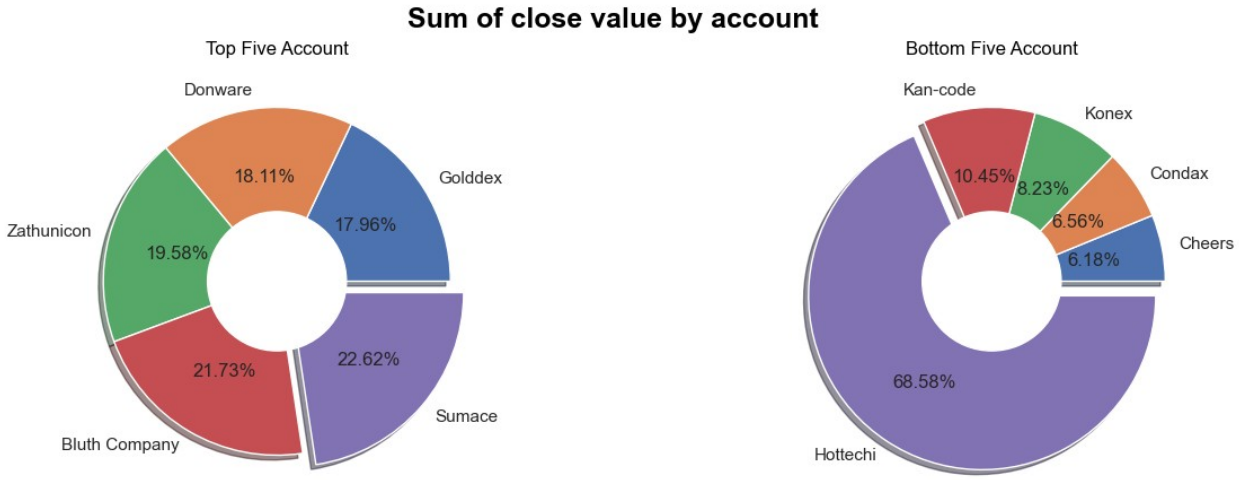
	account	close_value
29	Golddex	60577.493071
18	Donware	61109.746536
79	Zathunicon	66052.408583
6	Bluth Company	73303.070630
67	Sumace	76305.070630

```
y=sp.groupby("account")["close_value"].sum().reset_index()
y=y.sort_values("close_value",ascending=True).tail()
y
```

	account	close_value
9	Cheers	2.099473e+05
11	Condax	2.228101e+05
42	Konex	2.796814e+05
40	Kan-code	3.548732e+05
34	Hottechi	2.329948e+06

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.pie(x["close_value"],labels=x["account"],radius=1,shadow=True,auto
pct="%0.2f%%",explode=[0,0,0,0,0.1])
plt.pie([1],colors=['w'],radius=0.4)
plt.title("Top Five Account",color="black")
plt.subplot(1,2,2)
plt.pie(y["close_value"],labels=y["account"],radius=1,shadow=True,auto
pct="%0.2f%%",explode=[0,0,0,0,0.1])
plt.pie([1],colors=['w'],radius=0.4)
plt.title("Bottom Five Account",color="black")
```

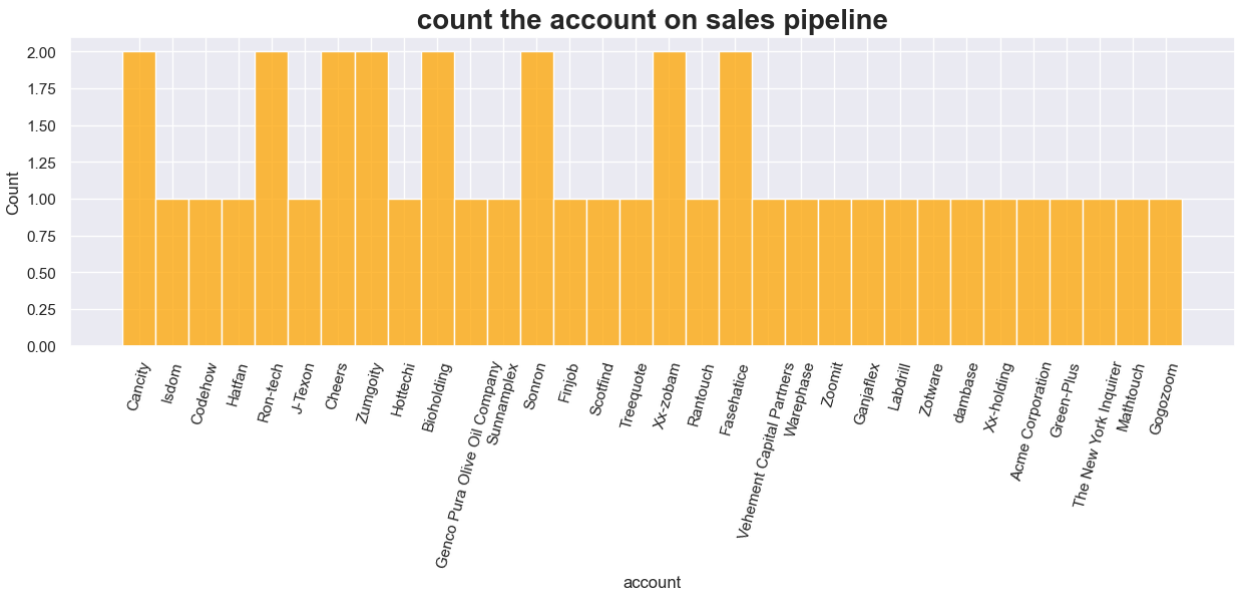
```
plt.suptitle("Sum of close value by
account",fontweight="bold",color="black",fontsize=18)
plt.show()
```




```
# 3.2:-count the total account on sales pipeline?
x=sp['account'].nunique()
x
```

```
85
```

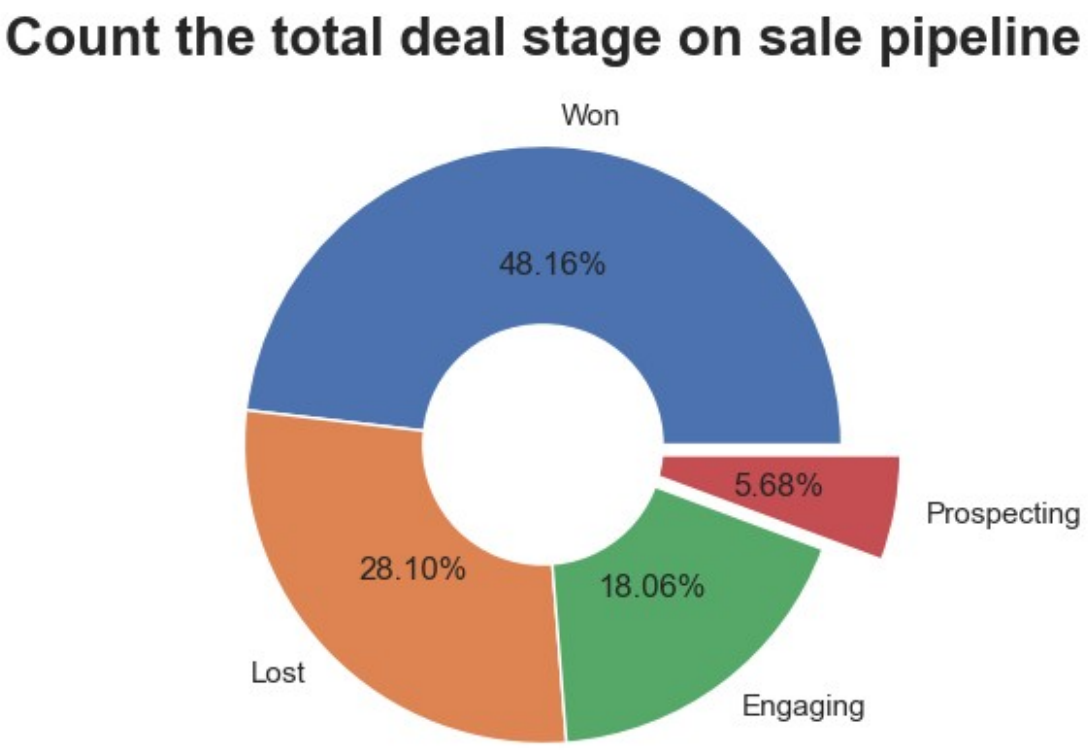
```
plt.figure(figsize=(15,4))
sns.histplot(data=sp.head(40),x="account",color="orange")
plt.title(" count the account on sales pipeline
",fontweight="bold",fontsize=20)
plt.xticks(rotation=75)
plt.show()
```



insights

```
#count the type of deal stage on sales pipeline
x=sp['deal_stage'].value_counts()
keys=x.keys()
val=x.values

plt.figure(figsize=(5,5))
plt.pie(val,labels=keys,autopct="%0.2f%%",explode=[0,0,0,0.2])
plt.pie([1],colors=['w'],radius=0.4)
plt.title("Count the total deal stage on sale
pipeline",fontweight="bold",fontsize=20)
plt.show()
```



insights

Deal Stage Overview

Title: Count the total deal stage on sale pipeline Data Breakdown:

Won: Percentage: 48.16% Lost: Percentage: 28.10% Engaging: Percentage: 18.06% Prospecting: Percentage: 5.68% Visual Representation:

The chart uses different colors to represent each stage: Won: Blue Lost: Orange Engaging: Green Prospecting: Red Summary:

The majority of the deals in the pipeline have been won, followed by lost deals, engaging, and prospecting stages.

Relationships and Trends Identified

#1. Sales Agent Performance Variation

#Some sales agents are closing significantly higher value deals than others.

#There's a clear disparity — a few agents might account for a majority of total revenue.

#This is a classic Pareto (80/20) Principle case — 20% of agents bringing in 80% of value.

`Trend:

#Top performers dominate deal closings.

#Lower performers might need training, mentorship, or better leads.

#2. Importance of Individual Contribution

#In a sales pipeline, individual success often drives team success.

#Tracking individual close values shows who drives business growth.

#Recognizing top agents can be key to motivating the team and setting benchmarks.

#3. Potential Need for Sales Strategy Adjustment

#If only a few agents are successful, it may indicate that:

#The lead distribution is uneven.

#Some agents are better at handling objections or qualifying leads.

#The sales process isn't standardized across the team.

General Trends Likely (based on sales pipeline data)

#If the other visuals are like common sales dashboards, they likely show:

#Deals by stage → Trend: Bottlenecks happen around "Negotiation" or "Proposal" stages.

#Conversion rates → Trend: Low conversion from early lead stages to final closing.

#Deal size vs. sales cycle → Trend: Larger deals take longer to close but yield more revenue.

Summary of Findings (so far)

High variance between sales agent performance.

#Top performers contribute disproportionately to total sales.

#Potential gaps in training, support, or opportunity distribution across the team.

#Strategic action (like coaching low performers, replicating best practices) could lift overall sales.

ENDING----------
