Emerton is a global high-end strategy consulting group specialities in Strategy consulting, Digital transformation, Business transformation, Innovation & Data analytics

Analyse and predict the win/bidding possibilities of deals/project for an IT consulting company and see how the possibility of bidding is impacted by other agents. It also will enable the company to manage the effort required to win the deal to meet the growth targets. To recommend top 5 Head-Bid managers

```
1
```

```
1
```

```
1
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
```

```
1 cd /content/drive/MyDrive/Colab_Notebooks/Learnbay_Interviews/WIN_Predcitio
```

```
/content/drive/MyDrive/Colab_Notebooks/Learnbay_Interviews/WIN_Predcitio
```

```
1 ls
```

```
Project_Win_Prediction_Analytics.pdf  WIN_Prediction_Finance.ipynb
Win_Prediction_Data.xlsx
```

```
1 # Importing the basic packages
2
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 import sklearn
```

```
import warnings
warnings.filterwarnings('ignore')
import os

import tensorflow as tf #tensor flow is the backend
from tensorflow import keras # keras is the frontend or wrapper; Keras is b
                             # 100 lines of code, In Keras it will be 5 lin
```

```
#importing the dataset

my_data = pd.read_excel('Win_Prediction_Data.xlsx')
my_data
```

	Client Category	Solution Type	Deal Date	Sector	Location	VP Name	
0	Telecom	Solution 7	2012-03-27	Sector 24	L5	Ekta Zutshi	-
1	Telecom	Solution 7	2012-09-25	Sector 24	L5	Ekta Zutshi	-
2	Internal	Solution 59	2011-08-01	Sector 20	Others	Ekta Zutshi	Russ
3	Internal	Solution 59	2011-04-28	Sector 20	Others	Ekta Zutshi	Russ
4	Internal	Solution 32	2011-06-03	Sector 20	Others	Ekta Zutshi	Russ
...	...	...	...	...	...	...	
10056	Power ind	Solution 9	2019-03-18	Sector 9	L5	Rudraksh Sharma	

```
newdata = my_data.copy()
newdata
```

	Client Category	Solution Type	Deal Date	Sector	Location	VP Name	
0	Telecom	Solution 7	2012-03-27	Sector 24	L5	Ekta Zutshi	-
1	Telecom	Solution 7	2012-09-25	Sector 24	L5	Ekta Zutshi	-
2	Internal	Solution 59	2011-08-01	Sector 20	Others	Ekta Zutshi	Russ
3	Internal	Solution 59	2011-04-28	Sector 20	Others	Ekta Zutshi	Russ
4	Internal	Solution 32	2011-06-03	Sector 20	Others	Ekta Zutshi	Russ

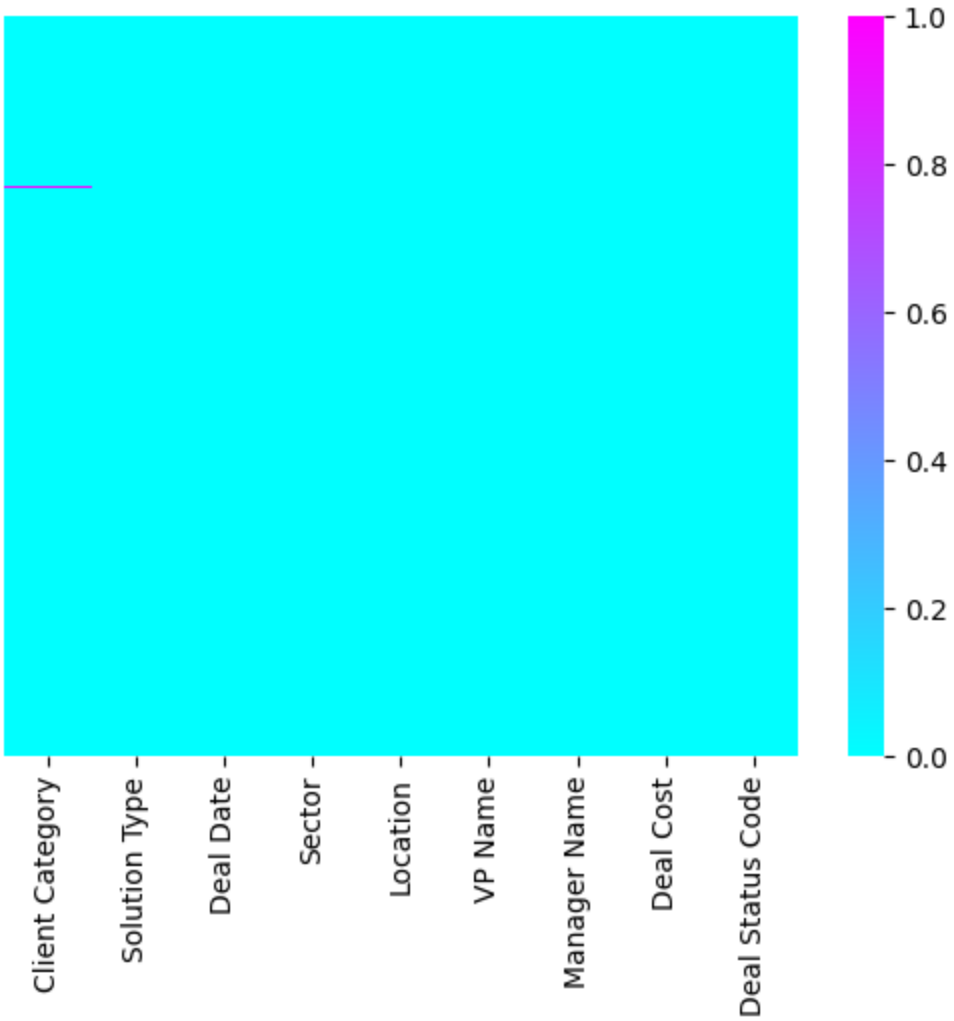
```
#Missing data

newdata.isnull().sum()
```

```
Client Category     79
Solution Type        0
Deal Date            0
Sector               0
Location             0
VP Name              0
Manager Name         0
Deal Cost            0
Deal Status Code     0
dtype: int64
```

```
newdata.isnull().sum()/len(newdata) * 100 # Missing percentage is .78%
```

```
Client Category     0.78521
Solution Type       0.00000
Deal Date           0.00000
Sector              0.00000
Location            0.00000
VP Name             0.00000
Manager Name        0.00000
Deal Cost           0.00000
Deal Status Code    0.00000
dtype: float64
```

```
1 sns.heatmap(newdata.isnull(), yticklabels = False, cbar=True, cmap='cool')
2 plt.show()
```




```
1   newdata.info()
2
3   # Most of the datatypes are object.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10061 entries, 0 to 10060
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Client Category   9982 non-null   object
 1   Solution Type     10061 non-null  object
 2   Deal Date         10061 non-null  datetime64[ns]
 3   Sector            10061 non-null  object
 4   Location          10061 non-null  object
```

```
 5   VP Name           10061 non-null  object
 6   Manager Name      10061 non-null  object
 7   Deal Cost         10061 non-null  float64
 8   Deal Status Code  10061 non-null  object
dtypes: datetime64[ns](1), float64(1), object(7)
memory usage: 707.5+ KB
```

```
# Handling Missing Values

Client_Category = newdata['Client Category'].value_counts()

# Maximum values are from Client Category of 1763
```

```
  #Imputinig the 'Client Category' field missing values with 'Others' class

newdata['Client Category'] = newdata['Client Category'].fillna('Others')
```

```
sns.heatmap(newdata.isnull(), yticklabels = False, cbar=True, cmap='cool')
plt.show()
```


```
# Describe the Object data type

sumcat = newdata.describe(include='O')
sumcat
```

	Client Category	Solution Type	Sector	Location	VP Name	Manager Name
count	10061	10061	10061	10061	10061	10061
unique	41	67	25	13	43	278
top	Others	Solution 32	Sector 23	L10	Mervin Harwood	Molly Eakes

```
newdata['Deal Status Code'].value_counts()

# Balanced dataset = (2 * Minority > Majority)
#                  = 2 * 3755 > 6306
#                  = 7510 > 6306  ( So its a Balanced dataset; so we need n


```

```
Lost    6306
Won     3755
Name: Deal Status Code, dtype: int64
```

```
Client_Category_index = newdata["Client Category"].value_counts()
Client_Category_index
```

```
Others                  1842
Internal                1454
Services_based          1202
Tech                     913
Domestic Public Bank     419
International Bank       376
Consulting               352
Finance                  339
Telecom                  327
Power ind                264
```

```
Domestic Private Bank     262
Insurance                 247
Consumer Good             185
Automobiles               178
Infrastructure            152
Domestic Bank             134
Retail_market             126
Govt                      121
Hospitality               119
Manufacturing             117
Pharma                    110
Healthcare                 99
Electronics                81
Media_Journal              71
Industries                 66
Research Development       63
Energy                     57
Knowledge                  50
Management                 43
Govt Bank Special          41
Payment                    40
Energy                     37
e-commerce                 32
Airpline                   27
Holding                    25
International Org          25
Logistics                  20
Real Estate                19
Share_market               14
Tax_audit                   7
Medical                     5
Name: Client Category, dtype: int64
```

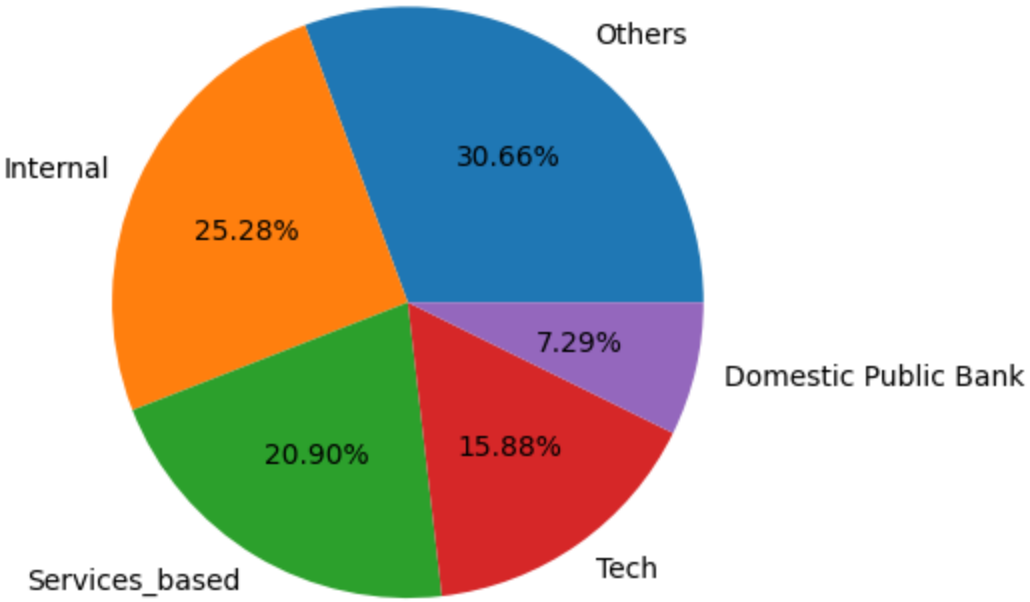
```
1 Client_Category_index = newdata["Client Category"].value_counts().index
2 Client_Category_index
```

```
Index(['Others', 'Internal', 'Services_based', 'Tech', 'Domestic Public
Bank',
       'International Bank', 'Consulting', 'Finance', 'Telecom', 'Power
ind',
       'Domestic Private Bank', 'Insurance', 'Consumer Good',
'Automobiles',
       'Infrastructure', 'Domestic Bank', 'Retail_market', 'Govt',
       'Hospitality', 'Manufacturing', 'Pharma', 'Healthcare',
'Electronics',
       'Media_Journal', 'Industries', 'Research Development', 'Energy',
       'Knowledge', 'Management', 'Govt Bank Special', 'Payment',
'Energy ',
```

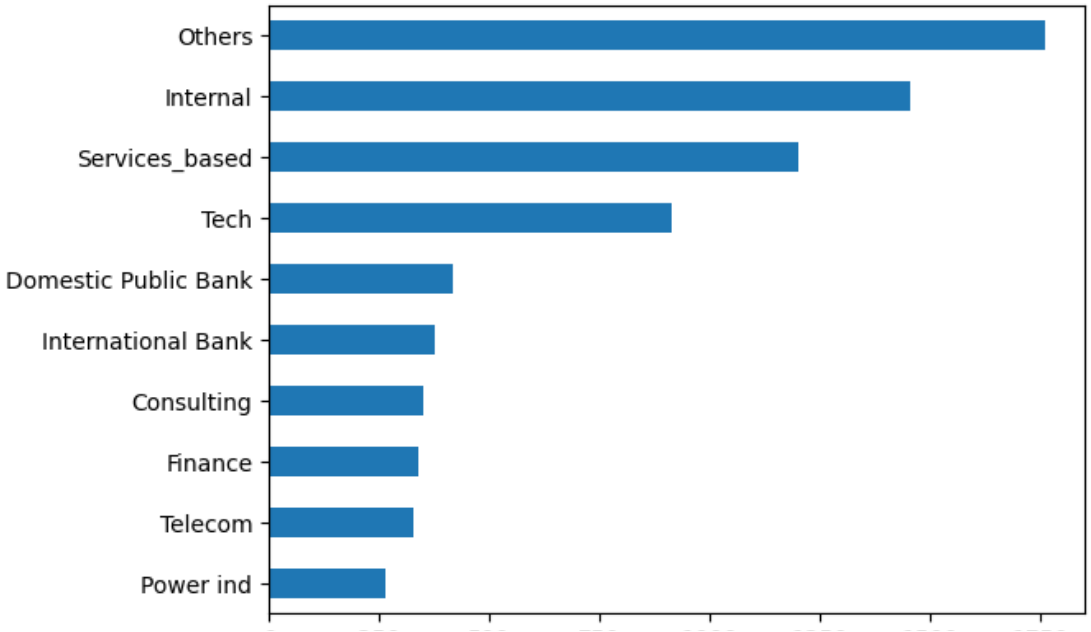
```
       'e-commerce', 'Airpline', 'Holding', 'International Org',
  'Logistics',
       'Real Estate', 'Share_market', 'Tax_audit', 'Medical '],
      dtype='object')
```

8. Recommending top 5 and 10 combination of SBU Head-Bid Manager.

```
plt.pie(Client_Category[:5], labels=Client_Category_index[:5], autopct = '%
plt.show()

# Pie chart is good for 5 values. If its more than 5 , then it has to be go
```




```
# If its more than for 10 categories, them Bar Graph will be better.

Client_Category[:10].plot(kind='barh').invert_yaxis()
```



Recommendation for top 5 and 10 Solution Type

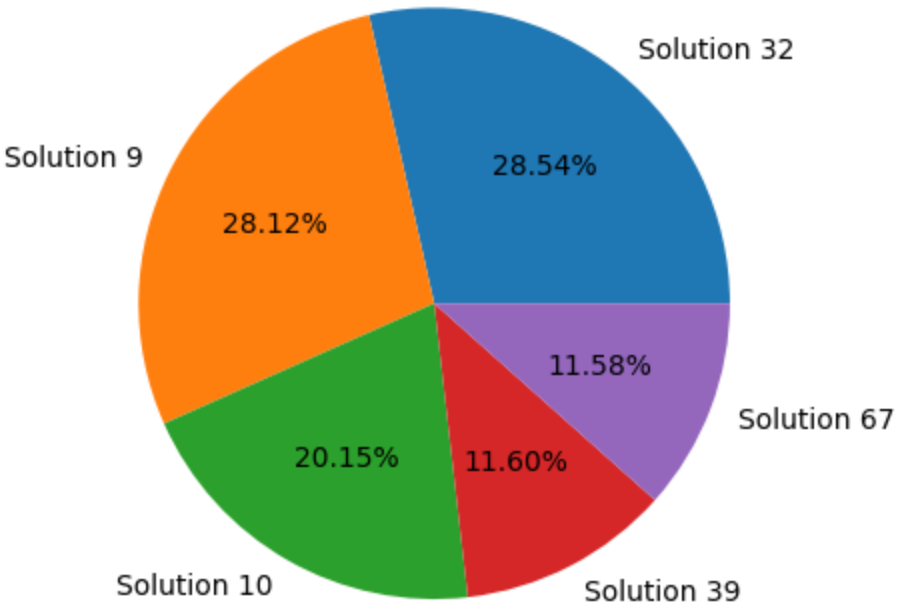
```
Solution_Type = newdata['Solution Type'].value_counts()

```

```
Solution_Type_index = newdata['Solution Type'].value_counts().index
Solution_Type_index
```

```
Index(['Solution 32', 'Solution 9', 'Solution 10', 'Solution 39',
       'Solution 67', 'Solution 37', 'Solution 59', 'Solution 12',
       'Solution 8', 'Solution 29', 'Solution 6', 'Solution 13',
'Solution 31',
       'Solution 4', 'Solution 28', 'Solution 38', 'Solution 7',
'Solution 11',
       'Solution 20', 'Solution 52', 'Solution 40', 'Solution 30',
       'Solution 14', 'Solution 36', 'Solution 58', 'Solution 47',
       'Solution 35', 'Solution 55', 'Solution 33', 'Solution 48',
       'Solution 16', 'Solution 26', 'Solution 49', 'Solution 2',
       'Solution 34', 'Solution 42', 'Solution 61', 'Solution 41',
       'Solution 65', 'Solution 44', 'Solution 17', 'Solution 15',
       'Solution 50', 'Solution 25', 'Solution 1', 'Solution 43',
       'Solution 24', 'Solution 62', 'Solution 5', 'Solution 27',
       'Solution 46', 'Solution 3', 'Solution 22', 'Solution 53',
       'Solution 51', 'Solution 45', 'Solution 63', 'Solution 23',
       'Solution 54', 'Solution 21', 'Solution 66', 'Solution 64',
       'Solution 60', 'Solution 57', 'Solution 56', 'Solution 18',
```

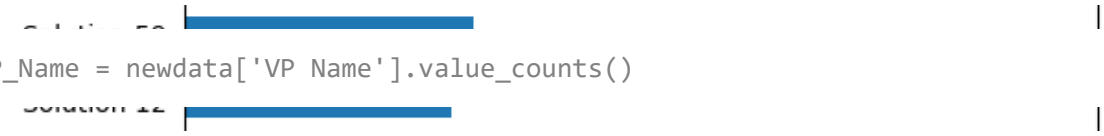
```
       'Solution 19'],
      dtype='object')
```

```
1 plt.pie(Solution_Type[:5], labels=Solution_Type_index[:5], autopct = '%1.2f
2 plt.show()
```




```
1 Solution_Type[:10].plot(kind='barh').invert_yaxis()
```

Recommendation for top 5 and 10 VP Name



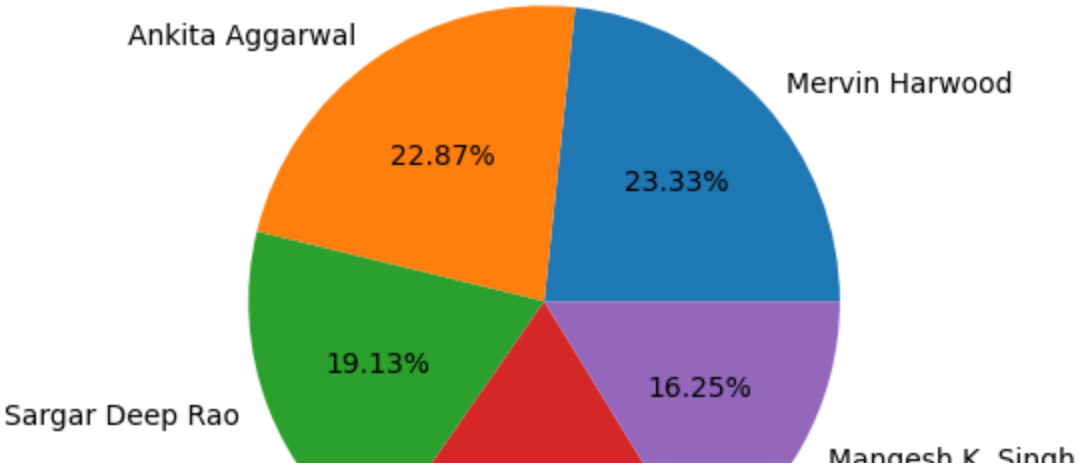
```
1 VP_Name = newdata['VP Name'].value_counts()
```



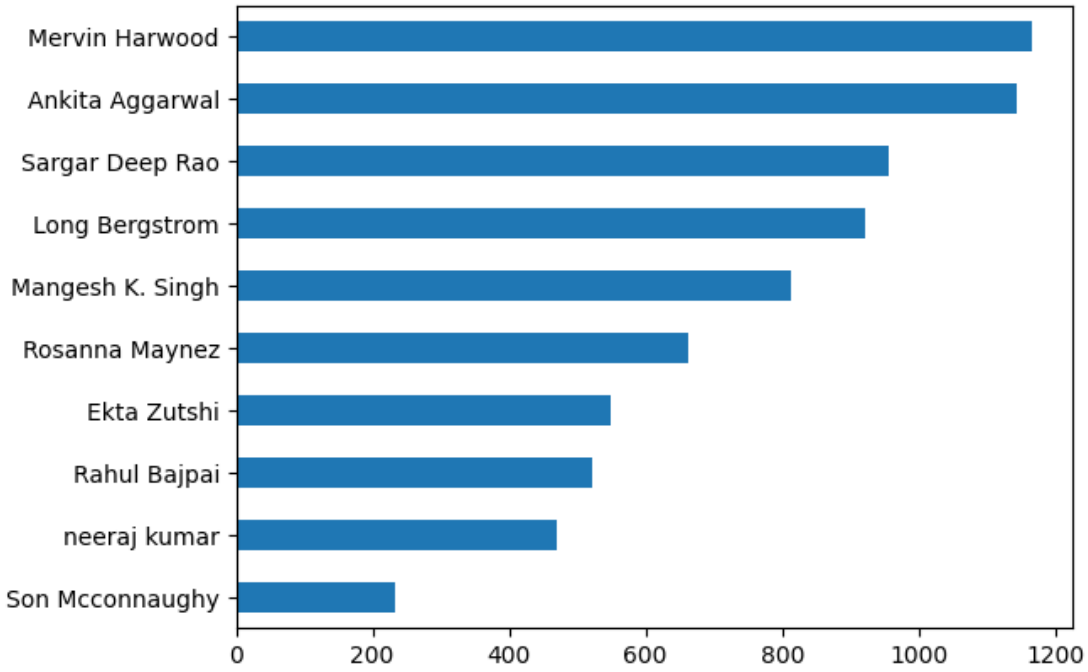
```
1 VP_Name_index = newdata['VP Name'].value_counts().index
2 VP_Name_index
```

```
Index(['Mervin Harwood', 'Ankita Aggarwal', 'Sargar Deep Rao',
       'Long Bergstrom', 'Mangesh K. Singh', 'Rosanna Maynez', 'Ekta
Zutshi',
       'Rahul Bajpai', 'neeraj kumar', 'Son Mcconnaughy', 'som dutt',
       'Hardeep Suksma', 'Brendon Wycoff', 'Clinton Mani', 'Saurabh
Singh',
       'Jewell Tunstall', 'Rudraksh Sharma', 'Gopa Trilochana',
       'Russell Dahlen', 'Molly Eakes', 'Alam Syed', 'Gaurav Sameria',
       'Varsha Arora', 'Gayle Molter', 'Vidur Hukle', 'Earline Langton',
       'Manpreet Singh', 'Mayank Mewar', 'Marcella Mo', 'Rahul Kocher',
       'Man Suddeth', 'Jitendra Choudhary', 'Prashant Rawat',
'Lilli Storrs',
       'Vikram Rawat', 'Kamelesh Srinivasan', 'Dennis Faux', 'Visvajeet
Das',
       'Waylon Mulder', 'P. Somya', 'Kirk Hofmeister', 'Sarthak Batra',
       'md. afsar'],
      dtype='object')
```

```
1 plt.pie(VP_Name[:5], labels=VP_Name_index[:5], autopct = '%1.2f%%')
2 plt.show()
```




```
1 VP_Name[:10].plot(kind='barh').invert_yaxis()
```



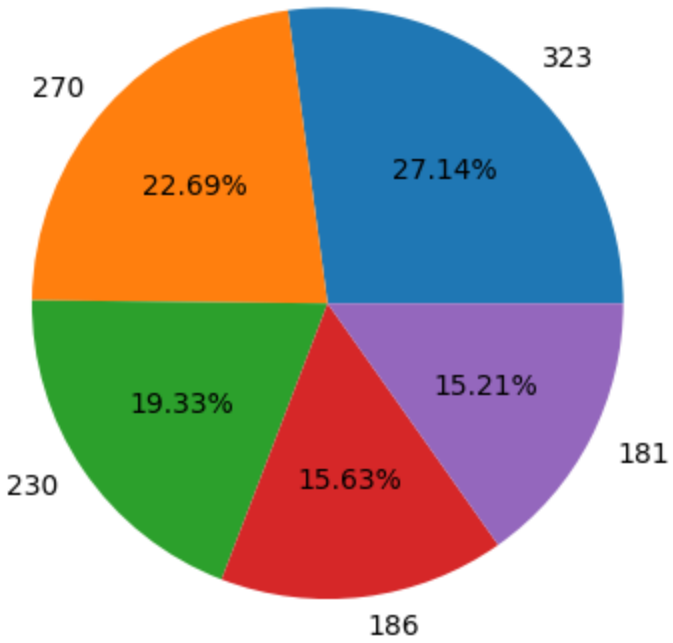
Recommendation for 5 and 10 Manager Name

```
Manager_Name = newdata['Manager Name'].value_counts()
```

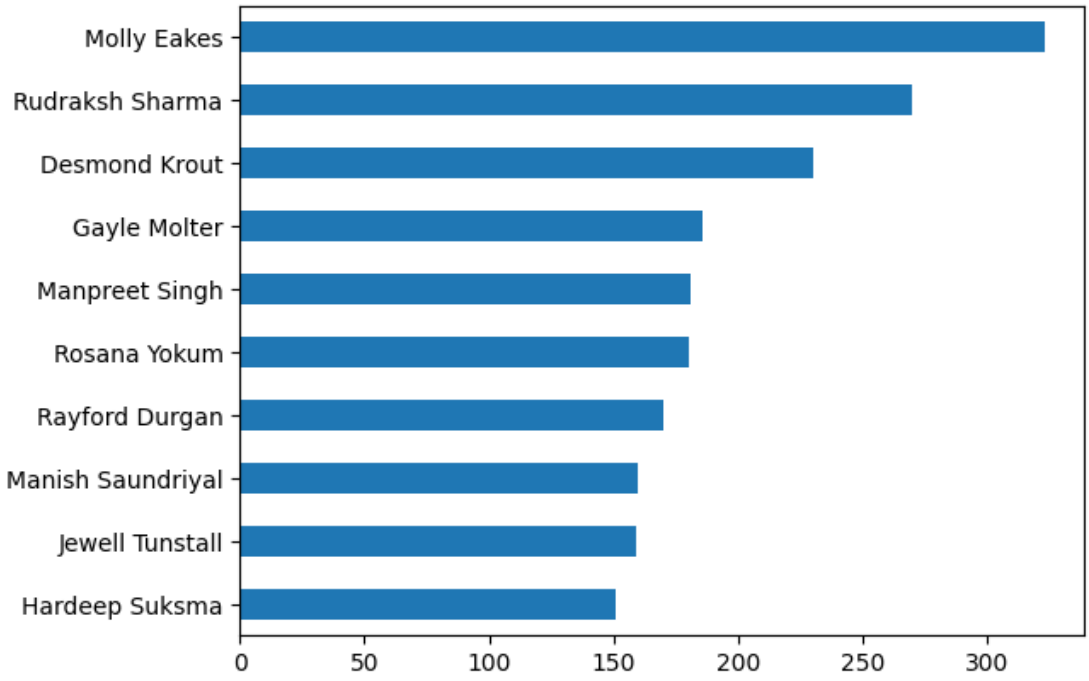
```
Manager_Name_index = newdata['Manager Name'].value_counts()
Manager_Name_index
```

```
Molly Eakes        323
Rudraksh Sharma    270
Desmond Krout      230
Gayle Molter       186
Manpreet Singh     181
                  ...
Anju Nanda           1
Taran Singh          1
pooran chand         1
Rishab Bhatt         1
Cleotilde Biron      1
Name: Manager Name, Length: 278, dtype: int64
```

```
plt.pie(Manager_Name[:5], labels=Manager_Name_index[:5], autopct = '%1.2f%%
plt.show()
```




```
Manager_Name[:10].plot(kind='barh').invert_yaxis()
```



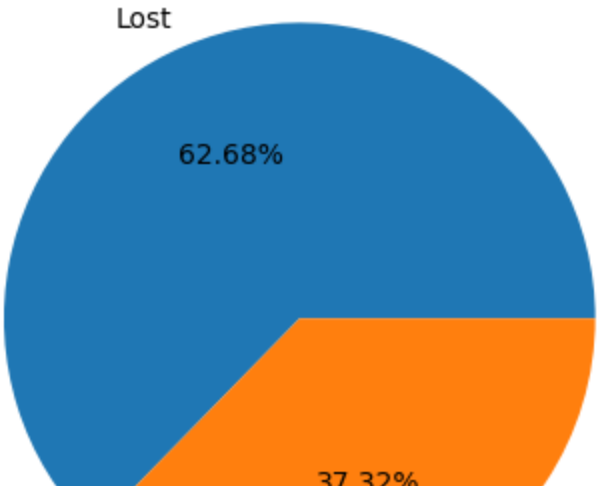
Deal Status Code

```
Deal_Status_code = newdata['Deal Status Code'].value_counts()
```

```
Deal_Status_Code_index = newdata["Deal Status Code"].value_counts().index
Deal_Status_Code_index
```

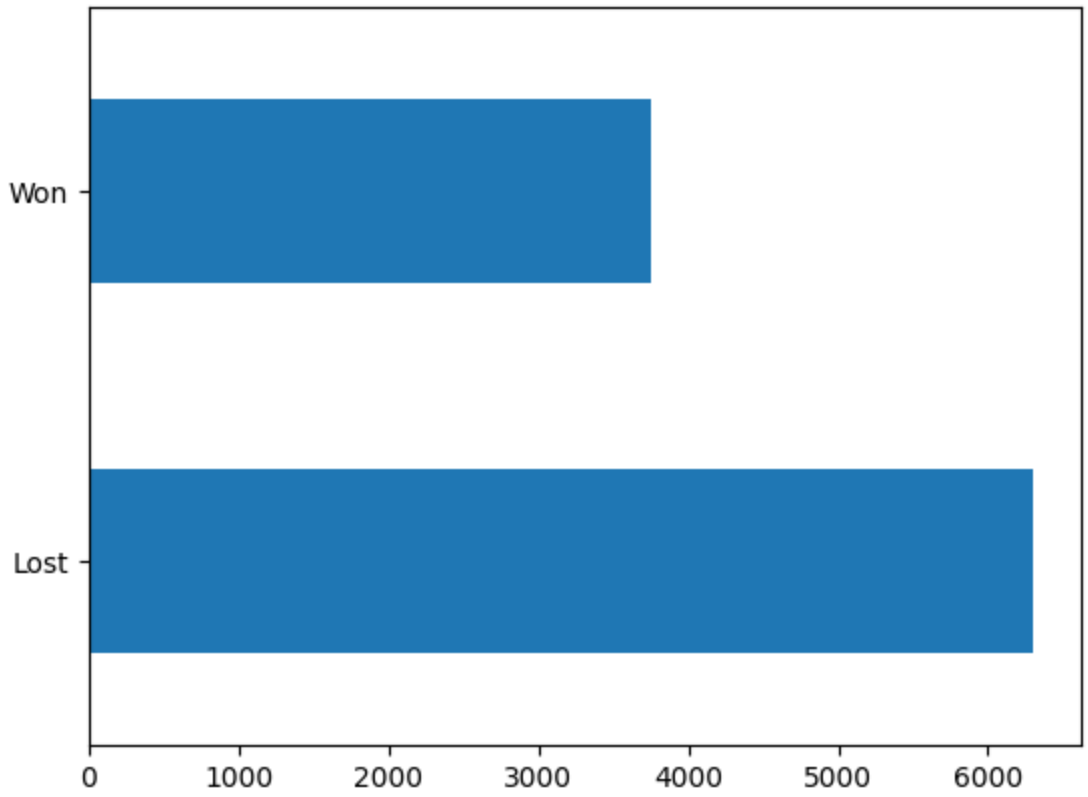
```
Index(['Lost', 'Won'], dtype='object')
```

```
plt.pie(Deal_Status_code[:5], labels=Deal_Status_Code_index[:5], autopct =
plt.show()
```




```
1 Deal_Status_code[:10].plot(kind='barh')
```

```
<Axes: >
```



Relationship between Independent Vs Dependent variables

```
1 newdata.columns
2
3 # Mainly we need relationships between Client Category, Solution type, VP n
```

```
Index(['Client Category', 'Solution Type', 'Deal Date', 'Sector',
'Location',
       'VP Name', 'Manager Name', 'Deal Cost', 'Deal Status Code'],
      dtype='object')
```

```
1 newdata.head(2)
```

	Client Category	Solution Type	Deal Date	Sector	Location	VP Name	Manager Name	Dea Cos
		Solution	2012-	Sector		Ekta	Gopa	

Relationship between Client Category and Deal Status Code

```
1 rel_client_cat = newdata[['Client Category', 'Deal Status Code']].groupby([
2
```

```
1 rel_client_cat
```

	Client Category	Deal Status Code	Deal Status Code
0	Airpline	Lost	22
1	Airpline	Won	5
2	Automobiles	Lost	112
3	Automobiles	Won	66
4	Consulting	Lost	182

Relationship between Solution Type and Deal Status Code

```
1 Solution_Type_cat = newdata[['Solution Type', 'Deal Status Code']].groupby(
```

79	Telecom	Won	135

```
1 Solution_Type_cat
```

	Solution Type	Deal Status Code	Deal Status Code
0	Solution 1	Lost	2
1	Solution 1	Won	3
2	Solution 10	Lost	690
3	Solution 10	Won	326
4	Solution 11	Lost	88
...	...	...	...
111	Solution 7	Won	62
112	Solution 8	Lost	190
113	Solution 8	Won	184
114	Solution 9	Lost	1018
115	Solution 9	Won	400

116 rows × 3 columns

Relationship between VP Name and Deal Status Code

```
1 VP_Name_cat = newdata[['VP Name', 'Deal Status Code']].groupby(['VP Name',
```

```
1 VP_Name_cat
```

	VP Name	Deal Status Code	Deal Status Code
0	Alam Syed	Lost	62
1	Alam Syed	Won	64
2	Ankita Aggarwal	Lost	866
3	Ankita Aggarwal	Won	277
4	Brendon Wycoff	Lost	132
...	...	...	...
77	md. afsar	Lost	2
78	neeraj kumar	Lost	254
79	neeraj kumar	Won	217
80	som dutt	Lost	138
81	som dutt	Won	82

82 rows × 3 columns

Relationship between Manager and Deal Status Code

```
1 Manager_Name_cat = newdata[['Manager Name', 'Deal Status Code']].groupby(['
2 Manager_Name_cat
```

	Manager Name	Deal Status Code	Total Count
0	Aastha Gandhi	Won	1
1	Abhinav Warrier	Lost	80
2	Abhinav Warrier	Won	15
3	Abhishek Singhal	Lost	44
4	Abhishek Singhal	Won	23

```
1 pd.pivot_table(newdata, index = 'Deal Status Code', columns = 'Manager Name
```

Manager Name	Aastha Gandhi	Abhinav Warrier	Abhishek Singhal	Abhiskek Kumar	Abhiskh
Deal Status Code					
Lost	NaN	588808.824000	772860.963182	949894.957857	440767.97
Won	242647.06	302235.296667	476956.522609	587507.740789	375658.82

2 rows × 278 columns

```
1 pd.pivot_table(newdata, index = 'Deal Status Code', columns = 'VP Name', va
```

VP Name	Alam Syed	Ankita Aggarwal	Brendon Wycoff	Clinton Mani	Denn

Which Year my Revenue is High

```
1 import datetime
2 newdata['Deal_Date_Year'] = newdata['Deal Date'].dt.year # For Year
3 # Like Year wise Report Card
```

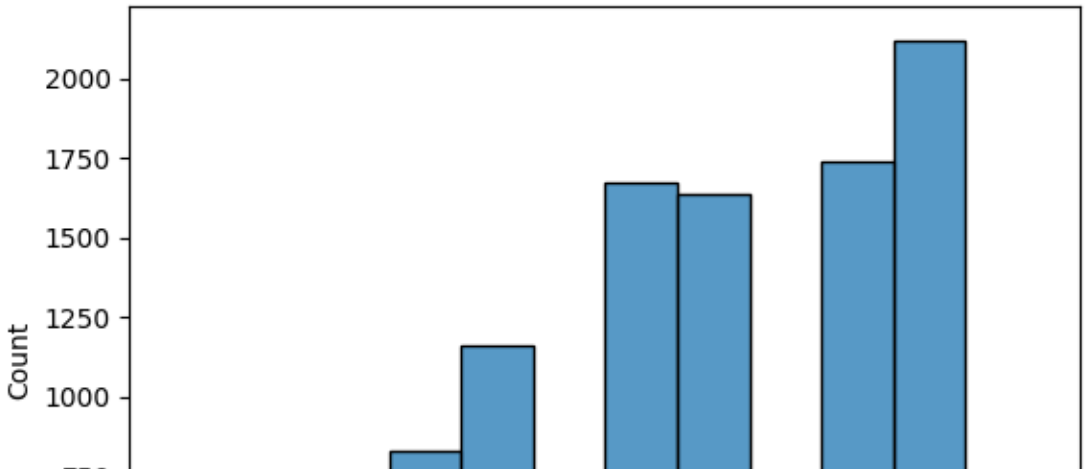
2 rows × 43 columns

```
1 newdata.head()
```

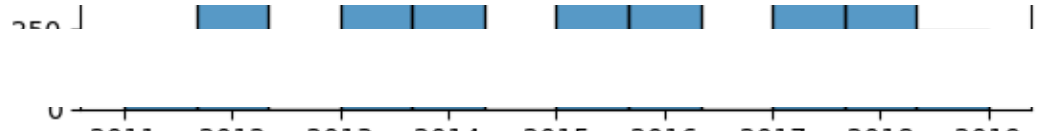
	Client Category	Solution Type	Deal Date	Sector	Location	VP Name	Manager Name	
0	Telecom	Solution 7	2012-03-27	Sector 24	L5	Ekta Zutshi	Gopa Trilochana	1500
1	Telecom	Solution 7	2012-09-25	Sector 24	L5	Ekta Zutshi	Gopa Trilochana	7447
		Solution	2011-	Sector		Ekta		

Yearly Data

```
1 sns.histplot(newdata.Deal_Date_Year, bins=12)
2 plt.show()
3
4 #Observations: From 2011 to 2019 there is a upward trend moving on.
```



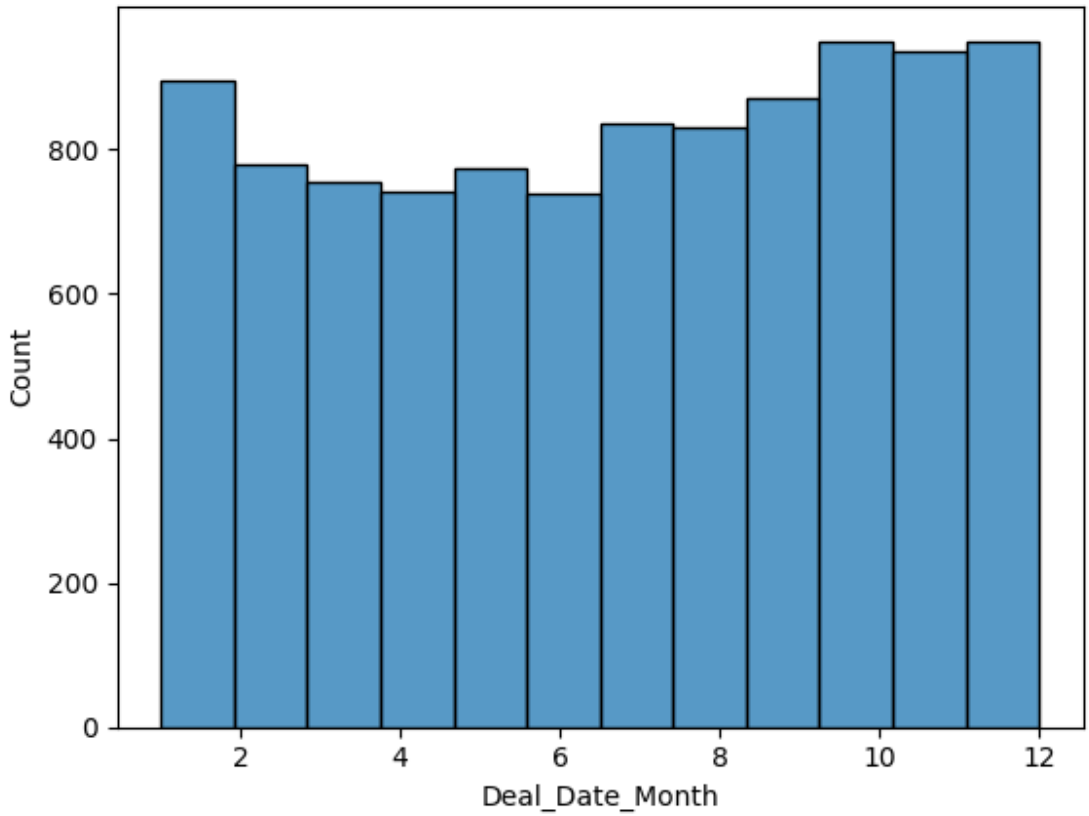
Monthly Data



```
1
1
1
1
1
1
1
1
1
```

```
1 newdata['Deal_Date_Month'] = newdata['Deal Date'].dt.month # For month
```

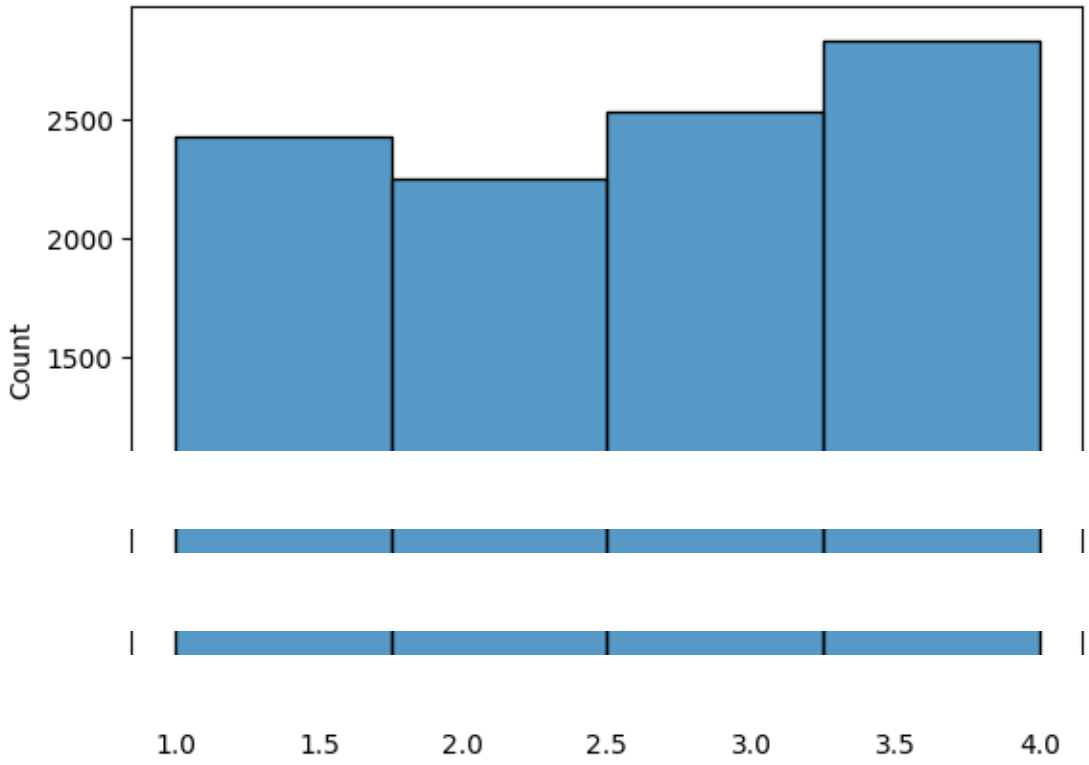
```
1 sns.histplot(newdata.Deal_Date_Month, bins=12)
2 plt.show()
```



Quarterly Data

```
1 newdata['Deal_Date_quarter'] = newdata['Deal Date'].dt.quarter # For Quarte
```

```
1 sns.histplot(newdata.Deal_Date_quarter, bins=4)
2 plt.show()
```



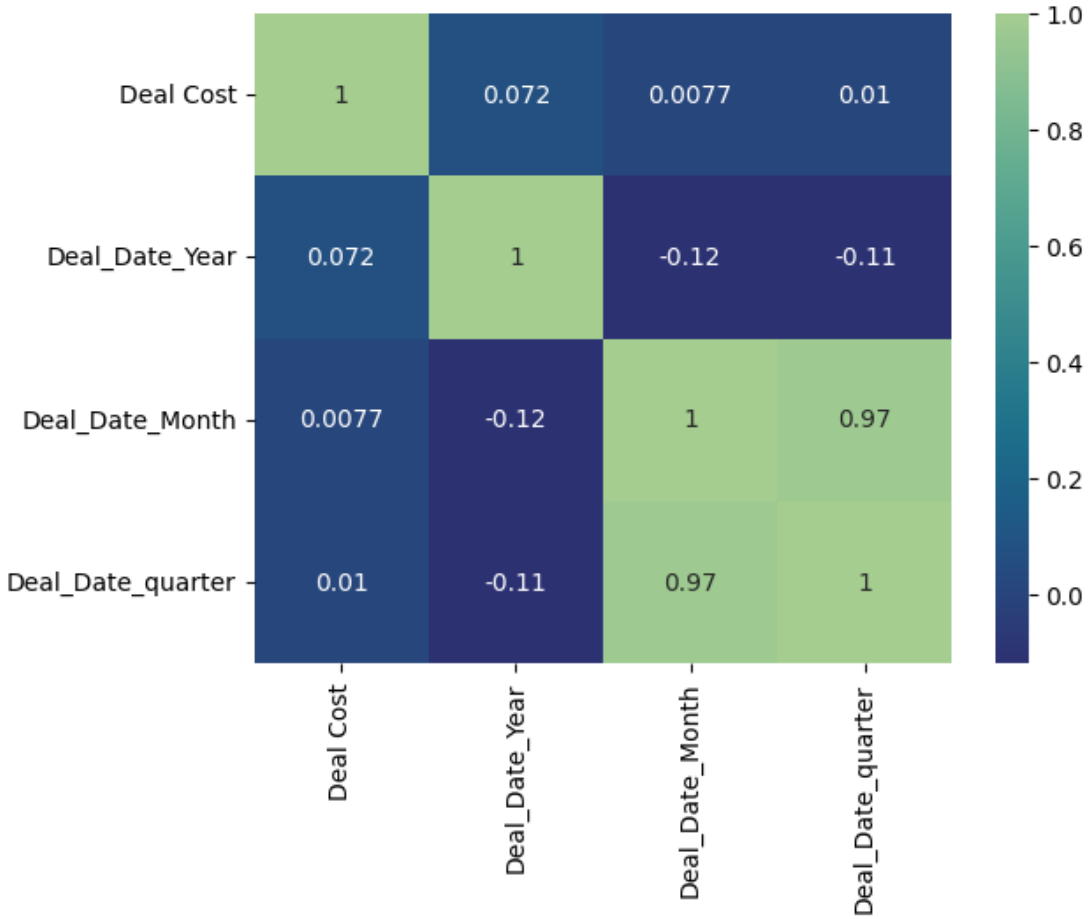
1
1
1
1
1
1
1
1
1

Heatmap:

for checking collinearity like which variable has the strong correlation

```
1 sns.heatmap(newdata.corr(), annot=True, cmap='crest_r')
```

```
<Axes: >
```




```
1 newdata.head(5)
```

	Client Category	Solution Type	Deal Date	Sector	Location	VP Name	Manager Name	
0	Telecom	Solution 7	2012-03-27	Sector 24	L5	Ekta Zutshi	Gopa Trilochana	1500

```
1 newdata['Client Category'].value_counts()
```

```
Others                  1842
Internal                1454
Services_based          1202
Tech                     913
Domestic Public Bank     419
International Bank       376
Consulting               352
Finance                  339
Telecom                  327
Power ind                264
Domestic Private Bank    262
Insurance                247
Consumer Good            185
Automobiles              178
Infrastructure           152
Domestic Bank            134
Retail_market            126
Govt                     121
Hospitality              119
Manufacturing            117
Pharma                   110
Healthcare                99
Electronics               81
Media_Journal             71
Industries                66
Research Development      63
Energy                    57
Knowledge                 50
Management                43
Govt Bank Special         41
Payment                   40
Energy                    37
e-commerce                32
Airpline                  27
Holding                   25
International Org         25
Logistics                 20
Real Estate               19
Share_market              14
Tax_audit                  7
```

```
Medical                     5
Name: Client Category, dtype: int64
```

```
newdata.columns
```

```
Index(['Client Category', 'Solution Type', 'Deal Date', 'Sector',
'Location',
       'VP Name', 'Manager Name', 'Deal Cost', 'Deal Status Code',
       'Deal_Date_Year', 'Deal_Date_Month', 'Deal_Date_quarter'],
      dtype='object')
```

```
newdata = newdata.drop(['Deal Date'], axis=1)
```

```
newdata.columns
```

```
Index(['Client Category', 'Solution Type', 'Sector', 'Location', 'VP
Name',
       'Manager Name', 'Deal Cost', 'Deal Status Code',
'Deal_Date_Year',
       'Deal_Date_Month', 'Deal_Date_quarter'],
      dtype='object')
```

```
newdata['Client Category'] = newdata['Client Category'].astype('category')
newdata['Client Category'] = newdata['Client Category'].cat.codes

newdata['Solution Type'] = newdata['Solution Type'].astype('category')
newdata['Solution Type'] = newdata['Solution Type'].cat.codes

newdata['Sector'] = newdata['Sector'].astype('category')
newdata['Sector'] = newdata['Sector'].cat.codes

newdata['Location'] = newdata['Location'].astype('category')
newdata['Location'] = newdata['Location'].cat.codes

newdata['VP Name'] = newdata['VP Name'].astype('category')
newdata['VP Name'] = newdata['VP Name'].cat.codes

newdata['Manager Name'] = newdata['Manager Name'].astype('category')
newdata['Manager Name'] = newdata['Manager Name'].cat.codes

newdata['Deal Status Code'] = newdata['Deal Status Code'].astype('category'
newdata['Deal Status Code'] = newdata['Deal Status Code'].cat.codes
```

```
1 newdata.head(5)
```

	Client Category	Solution Type	Sector	Location	VP Name	Manager Name	Deal Cost	Deal Status Code
0	39	64	16	7	6	82	150000.00	1
1	39	64	16	7	6	82	744705.88	1
2	19	54	12	12	6	183	60000.00	0
3	19	54	12	12	6	183	60000.00	0

```
1 # We can remove Deal_Date_Year, Deal_Date_Month, Deal_Date_quarter as these
2
3 newdata = newdata.iloc[:,0:8]
```

```
1 newdata
```

	Client Category	Solution Type	Sector	Location	VP Name	Manager Name	Deal Cost	S
0	39	64	16	7	6	82	150000.00	
1	39	64	16	7	6	82	744705.88	
2	19	54	12	12	6	183	60000.00	
3	19	54	12	12	6	183	60000.00	
4	19	25	12	12	6	183	80882.35	
...	...	...	...	...	...	...	...	
10056	31	66	24	7	29	182	588235.29	
10057	19	55	12	12	29	198	777058.82	
10058	31	66	24	7	29	182	588235.29	
10059	31	58	24	7	19	50	3042058.82	
10060	28	66	3	1	34	216	147058.82	

Split the data into X and Y

```
x = newdata.iloc[:,0:-1].values
y = newdata['Deal Status Code'].values
```

Feature Scaling:

Since the 'Deal Cost' column is too much compared to other values, we need to do 'Feature Scaling'.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x1 = sc.fit_transform(x)
```

```
x1
```

```
array([[ 1.40299487,  1.29765094,  0.78535088, ..., -1.11119992,
        -0.71544412, -0.38084443],
       [ 1.40299487,  1.29765094,  0.78535088, ..., -1.11119992,
        -0.71544412, -0.01369972],
       [-0.33339304,  0.86106354,  0.01663725, ..., -1.11119992,
         0.60004268, -0.43640639],
       ...,
       [ 0.70843971,  1.38496843,  2.32277814, ...,  0.79154462,
         0.58701806, -0.11029764],
       [ 0.70843971,  1.0356985 ,  2.32277814, ..., -0.03573561,
        -1.13223202,  1.40458284],
       [ 0.44798152,  1.38496843, -1.71296842, ...,  1.20518474,
         1.0298552 , -0.38266018]])
```

Splitting the Data into Train & Test

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x1, y, train_size=0.70,
```

Deep Neural Network 1:30

```
import tensorflow as tf
from tensorflow import keras
```

```
# Architecture - 1) Sequential (If its going in sequence) 2) Functional (No

dnn = tf.keras.models.Sequential()  # API
dnn.add(tf.keras.layers.Dense(units=20, activation='relu'))   # 1st Hidden
dnn.add(tf.keras.layers.Dense(units=20, activation='relu'))   # 2nd Hidden
dnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))  # Output ==>
dnn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accurac
dnn.fit(x_train, y_train, batch_size=32, epochs=100) # batch_size means spl

# Accuracy keep on increasing. This is called back propagation # At the 100
```

```
221/221 [==============================] - 0s 2ms/step - loss: 0.6015
Epoch 84/100
221/221 [==============================] - 0s 2ms/step - loss: 0.6014
Epoch 85/100
221/221 [==============================] - 0s 2ms/step - loss: 0.6020
Epoch 86/100
221/221 [==============================] - 0s 2ms/step - loss: 0.6014
Epoch 87/100
221/221 [==============================] - 0s 2ms/step - loss: 0.6019
Epoch 88/100
221/221 [==============================] - 0s 2ms/step - loss: 0.6014
Epoch 89/100
221/221 [==============================] - 0s 2ms/step - loss: 0.6003
Epoch 90/100
221/221 [==============================] - 1s 3ms/step - loss: 0.6002
Epoch 91/100
221/221 [==============================] - 1s 2ms/step - loss: 0.6002
Epoch 92/100
221/221 [==============================] - 1s 2ms/step - loss: 0.6013
Epoch 93/100
221/221 [==============================] - 1s 2ms/step - loss: 0.5998
Epoch 94/100
221/221 [==============================] - 0s 2ms/step - loss: 0.5985
Epoch 95/100
221/221 [==============================] - 1s 2ms/step - loss: 0.5981
Epoch 96/100
221/221 [==============================] - 0s 2ms/step - loss: 0.5984
Epoch 97/100
221/221 [==============================] - 1s 2ms/step - loss: 0.5983
Epoch 98/100
221/221 [==============================] - 1s 2ms/step - loss: 0.5986
Epoch 99/100
221/221 [==============================] - 0s 2ms/step - loss: 0.5981
Epoch 100/100
```

To Improve the accuracy, we are increasing the count as epochs=50, the best part is it will restart from where it left (ex: accuracy: 0.6779); & not from the beginning.

```
1 dnn.fit(x_train, y_train, batch_size=32, epochs=50)
```

```
Epoch 26/50
221/221 [==============================] - 1s 3ms/step - loss: 0.5906
Epoch 27/50
221/221 [==============================] - 1s 4ms/step - loss: 0.5905
Epoch 28/50
221/221 [==============================] - 1s 3ms/step - loss: 0.5900
```

```
1 dnn.fit(x_train, y_train, batch_size=32, epochs=50)
```

```
221/221 [==============================] - 0s 2ms/step - loss: 0.5802
Epoch 23/50
```

```
Epoch 45/50
221/221 [==============================] - 0s 2ms/step - loss: 0.5782
Epoch 46/50
221/221 [==============================] - 0s 2ms/step - loss: 0.5786
Epoch 47/50
221/221 [==============================] - 0s 2ms/step - loss: 0.5778
Epoch 48/50
221/221 [==============================] - 0s 2ms/step - loss: 0.5775
Epoch 49/50
221/221 [==============================] - 0s 2ms/step - loss: 0.5782
Epoch 50/50
```

```
1 y_pred = dnn.predict(x_test)
2 y_pred = (y_pred > 0.5)
3 y_pred
```

```
95/95 [==============================] - 0s 1ms/step
array([[False],
       [ True],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

```
1 from sklearn.metrics  import confusion_matrix, classification_report, accur
```

```
1 print(confusion_matrix(y_test, y_pred))
```

```
[[1470  485]
 [ 557  507]]
```

```
1 print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.75      0.74      1955
           1       0.51      0.48      0.49      1064

    accuracy                           0.65      3019
   macro avg       0.62      0.61      0.62      3019
weighted avg       0.65      0.65      0.65      3019
```

```
1 print(accuracy_score(y_test, y_pred)) # so we need improvement as the accur
```

```
0.6548526001987413
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

✓ 0s completed at 3:00 PM