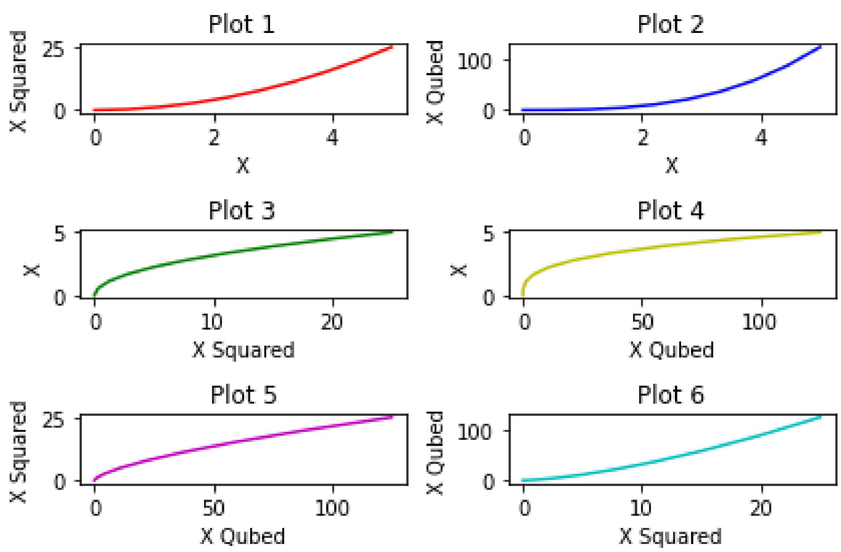
PDV- SECOND SESSIONALS

1. Write python code to create subplot containing three rows. Each row should contain two graphs. Type of visualization need to be created is line graph. Use different colors to represent lines in each graph. Provide proper labels and line ticks. (TLO 3.1)

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
fig, ax = plt.subplots(3, 2)

x = np.linspace(0,5,10)
x2 = x**2
x3 = x**3

ax[0,0].set_title('Plot 1')
ax[0,0].set_xlabel('X')
ax[0,0].set_ylabel('X Squared')

ax[0,0].plot(x,x2,'r')

ax[0,1].set_title('Plot 2')
ax[0,1].set_xlabel('X')
ax[0,1].set_ylabel('X Qubed')
ax[0,1].plot(x,x3,'b')

ax[1,0].set_title('Plot 3')
ax[1,0].set_xlabel('X Squared')
ax[1,0].set_ylabel('X ')
ax[1,0].plot(x2,x,'g')

ax[1,1].set_title('Plot 4')
ax[1,1].set_xlabel('X Qubed')
ax[1,1].set_ylabel('X ')
ax[1,1].plot(x3,x,'y')

ax[2,0].set_title('Plot 5')
ax[2,0].set_xlabel('X Qubed')
ax[2,0].set_ylabel('X Squared')
```

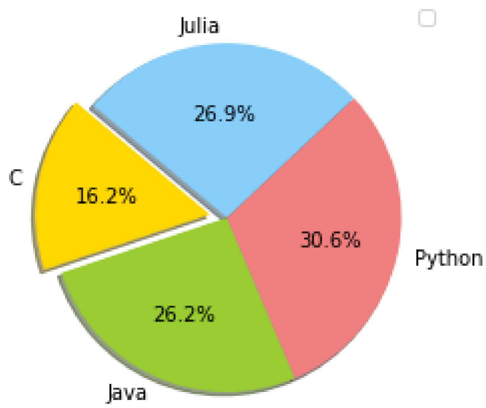
```
ax[2,0].plot(x3,x2,'m')

ax[2,1].set_title('Plot 6')
ax[2,1].set_xlabel('X Squared')
ax[2,1].set_ylabel('X Qubed')
ax[2,1].plot(x2,x3,'c')

fig.tight_layout()
```



2. Write a python code to create a pie chart. Generate random data. Visualization should include legend, user defined colors for each part of the pie chart. (TLO 3.2)

```
import matplotlib.pyplot as plt


# Data to plot
labels = 'C', 'Java', 'Python', 'Julia'
sizes = [130, 210, 245, 215]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0) # explode 1st slice



# Plot
Sections = plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.legend(Sections, labels, loc='best')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Legend does
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-add
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Legend does
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-add
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Legend does
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-add
  after removing the cwd from sys.path.
```



3. What is Exploratory Data Analysis(EDA)? Illustrate various steps performed in EDA with a reasoning for the same? specify atleast 5 EDA library. (TLO 2.3)

Exploratory Data Analysis: Exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

Steps to Perform in EDA

1)Variable Identification

2)Univariate Analysis

3)Bi-variate Analysis

4)Missing values treatment

5)Outlier treatment

6)Variable transformation

7)Variable creation

1)Variable identification: The very first step in exploratory data analysis is to identify the type of variables in the dataset. Variables are of two types — Numerical and Categorical. They can be further classified as variable, numerical,categorical,discrete, continuous etc. Let us take any dataset as example in that indentify the type of variable the next step is to identify the Predictor (Inputs) and Target (output) variables. Next steps are Importing the Librairies Importing Data Sets Identifying Data Sets

2)Univariate Analysis: Uni-variate analysis will depend on whether the variable type is categorical or continuous. Continuous Variables:- In case of continuous variables, we need to understand the central tendency and spread of the variable. Categorical Variables:- For categorical variables, we'll use frequency table to understand distribution of each category. We can also read as percentage of values under each category. It can be be measured using two metrics, Count and Count% against each category. Bar chart can be used as visualization.

3)Bi-variate Analysis: Bi-variate Analysis finds out the relationship between two variables. Here, we look for association and disassociation between variables at a pre-defined significance level. We can perform bi-variate analysis for any combination of categorical and continuous variables. The combination can be: Categorical & Categorical, Categorical & Continuous and Continuous & Continuous.

4)Missing values treatment: Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

EDA Automation Libararies are: pandas-profiling (using python)

D-Tale (using python)

sweetviz (using python)

autoviz (using python)

summarytools (using R)

explore (using R)

dataMaid (using R)

4. Write a Pandas program to split a dataset to group by two columns and then sort the aggregated results within the groups. The dataset consists of ord_no purch_amt ord_date customer_id salesman_id, group on 'customer_id',

'salesman_id' and then sort sum of purch_amt within the groups (TLO 2.3)

```
import pandas as pd
pd.set_option('display.max_rows', None)
df = pd.DataFrame({
'ord_no':[60001,60009,60002,60004,60007,60005,60008,60010,60003,60012,60011,60013],
'purch_amt':[180.5,260.65,85.26,100.5,928.5,2300.6,4560,1893.43,2480.4,250.45, 75.29,3045.6],
'ord_date': ['2020-10-05','2020-09-10','2020-10-05','2020-08-17','2020-09-10','2020-07-27','2
'customer_id':[1001,1001,1002,1001,1002,1001,1002,1001,1002,1001,1002,1002],
'salesman_id': [2002,2005,2001,2003,2002,2001,2001,2006,2003,2002,2007,2001]})
print("Original Orders DataFrame:")
print(df)
df_agg = df.groupby(['customer_id','salesman_id']).agg({'purch_amt':sum})
result = df_agg['purch_amt'].groupby(level=0, group_keys=False)
print("\nGroup on 'customer_id', 'salesman_id' and then sort sum of purch_amt within the grou
print(result.nlargest())
```

```
Original Orders DataFrame:
    ord_no  purch_amt    ord_date  customer_id  salesman_id
0    60001     180.50  2020-10-05         1001         2002
1    60009     260.65  2020-09-10         1001         2005
2    60002      85.26  2020-10-05         1002         2001
3    60004     100.50  2020-08-17         1001         2003
4    60007     928.50  2020-09-10         1002         2002
5    60005    2300.60  2020-07-27         1001         2001
6    60008    4560.00  2020-09-10         1002         2001
7    60010    1893.43  2020-10-10         1001         2006
8    60003    2480.40  2020-10-10         1002         2003
9    60012     250.45  2020-06-27         1001         2002
10   60011      75.29  2020-08-17         1002         2007
11   60013    3045.60  2020-04-25         1002         2001

Group on 'customer_id', 'salesman_id' and then sort sum of purch_amt within the groups:
customer_id  salesman_id
1001         2001           2300.60
             2006           1893.43
             2002            430.95
             2005            260.65
             2003            100.50
1002         2001           7690.86
             2003           2480.40
             2002            928.50
             2007             75.29
Name: purch_amt, dtype: float64
```

5. Write a Pandas program to replace the missing values with the most frequent values present in each column of a given dataframe. Test data consists of following attributes: ord_no, purch_amt, sale_amt, ord_date, customer_id, salesman_id. (TLO 2.2)

```
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
df = pd.DataFrame({
'ord_no':[60001,np.nan,60002,60004,np.nan,60005,np.nan,60010,60003,60012,np.nan,60013],
'purch_amt':[180.5,np.nan,85.26,100.5,928.5,np.nan,4560,1983.43,np.nan,250.45, 75.29,3045.6],
'sale_amt':[10.5,20.65,np.nan,11.5,98.5,np.nan,57,19.43,np.nan,25.45, 75.29,35.6],
'ord_date': ['2020-10-05','2020-09-10',np.nan,'2020-08-17','2020-09-10','2020-07-27','2020-09
'customer_id':[1002,1001,1001,1003,1002,1001,1001,1004,1003,1002,1001,1001],
'salesman_id':[2002,2003,2001,np.nan,2002,2001,2001,np.nan,2003,2002,2003,np.nan]})
print("Original Orders DataFrame:")
print(df)
print("\nReplace the missing values with the most frequent values present in each column:")
result = df.fillna(df.mode().iloc[0])
print(result)
```

```
Original Orders DataFrame:
     ord_no  purch_amt  sale_amt    ord_date  customer_id  salesman_id
0   60001.0     180.50     10.50  2020-10-05         1002       2002.0
1       NaN        NaN     20.65  2020-09-10         1001       2003.0
2   60002.0      85.26       NaN         NaN         1001       2001.0
3   60004.0     100.50     11.50  2020-08-17         1003          NaN
4       NaN     928.50     98.50  2020-09-10         1002       2002.0
5   60005.0        NaN       NaN  2020-07-27         1001       2001.0
6       NaN    4560.00     57.00  2020-09-10         1001       2001.0
7   60010.0    1983.43     19.43  2020-10-10         1004          NaN
8   60003.0        NaN       NaN  2020-10-10         1003       2003.0
9   60012.0     250.45     25.45  2020-06-27         1002       2002.0
10      NaN      75.29     75.29  2020-08-17         1001       2003.0
11  60013.0    3045.60     35.60  2020-04-25         1001          NaN

Replace the missing values with the most frequent values present in each column:
     ord_no  purch_amt  sale_amt    ord_date  customer_id  salesman_id
0   60001.0     180.50     10.50  2020-10-05         1002       2002.0
1   60001.0      75.29     20.65  2020-09-10         1001       2003.0
2   60002.0      85.26     10.50  2020-09-10         1001       2001.0
3   60004.0     100.50     11.50  2020-08-17         1003       2001.0
4   60001.0     928.50     98.50  2020-09-10         1002       2002.0
5   60005.0      75.29     10.50  2020-07-27         1001       2001.0
6   60001.0    4560.00     57.00  2020-09-10         1001       2001.0
```

7	60010.0	1983.43	19.43	2020-10-10	1004	2001.0
8	60003.0	75.29	10.50	2020-10-10	1003	2003.0
9	60012.0	250.45	25.45	2020-06-27	1002	2002.0
10	60001.0	75.29	75.29	2020-08-17	1001	2003.0
11	60013.0	3045.60	35.60	2020-04-25	1001	2001.0