DVA Assignment 1

June 17, 2024

Q1. Write a program to create a DataFrame have E-commerce data and perform selection of row/column using loc() and iloc()

[1]:

```
import pandas as pd
import numpy as np
data = {
 'Product_Id': np.arange(1, 11),
 'Price': np.random.randint(500, 2000, 10),
 'Quantity' : np.random.randint(1, 10, 10),
 'Customer_Id' : np.random.randint(1101, 1200, 10)
}
df = pd.DataFrame(data)
print(df)
data_iloc = df.iloc[1:4, 2:4]
print("\n [iloc] Selecting rows from 1 to 3 and columns from 2 to 3\n",␣
 ↪data_iloc)
data_loc = df.loc[1:4]
print("\n [loc] Selecting rows from 1 to 4\n", data_loc)
```

```
   Product_Id  Price  Quantity  Customer_Id
0           1   1708         8         1156
1           2   1320         5         1115
2           3   1940         8         1171
3           4   1080         8         1186
4           5   1829         9         1159
5           6   1215         7         1156
6           7    901         6         1124
7           8    598         7         1101
8           9    707         8         1180
9          10    791         9         1171

 [iloc] Selecting rows from 1 to 3 and columns from 2 to 3
    Quantity  Customer_Id
1         5         1115
2         8         1171
3         8         1186

 [loc] Selecting rows from 1 to 4
    Product_Id  Price  Quantity  Customer_Id
```

```
1            2   1320          5          1115
2            3   1940          8          1171
3            4   1080          8          1186
4            5   1829          9          1159
```

Q2. Create a Series object S5 containing numbers. Write a program to store the square of the series values in object S6. Display S6's values which are >15.

[3]:

```
import pandas as pd
s5= pd.Series(np.random.randint(1,10, 5))
print(s5)
s6 = s5**2
print(s6)
new_series = s6[s6 > 15]
print(new_series)
```

```
0    1
1    7
2    5
3    1
4    6
dtype: int32
0     1
1    49
2    25
3     1
4    36
dtype: int32
1    49
2    25
4    36
dtype: int32
```

Q3. Write a program to fill all missing values in a DataFrame with zero

[4]:

```
data = {
 'A': [10, 24, np.nan, 41],
 'B': [65, np.nan, 27, 38],
 'C': [9, 10, 31, np.nan] }
df = pd.DataFrame(data)
print(df, "\n")
df_filled = df.fillna(0)
print(df_filled)
```

```
      A     B     C
0  10.0  65.0   9.0
1  24.0   NaN  10.0
2   NaN  27.0  31.0
3  41.0  38.0   NaN
```

```
      A     B     C
0  10.0  65.0   9.0
1  24.0   0.0  10.0
2   0.0  27.0  31.0
3  41.0  38.0   0.0
```

Q4. Program for combining DataFrames using concat(), join(),merge()

[6]:

```
import pandas as pd
data1= pd.DataFrame({
 'id' : [10, 11, 12, 13],
 'name' : ['Sourav', 'Karthik', 'Suryadev', 'Shakti'],
})
data2= pd.DataFrame({
 'id': [12, 13, 14],
 'age' : [20, 24, 31]
})
concat_df = pd.concat([data1, data2])
print("CONCATENATED DATAFRAME (vertically) \n", concat_df)
joined_df = data1.set_index('id').join(data2.set_index('id'), how='inner')
print("\n JOIN DATAFRAME \n", joined_df)
merge_df = pd.merge(data1, data2, on='id')
print("\n MERGE DATAFRAME \n", merge_df)
```

```
CONCATENATED DATAFRAME (vertically)
    id      name   age
0  10    Sourav   NaN
1  11   Karthik   NaN
2  12  Suryadev   NaN
3  13    Shakti   NaN
0  12       NaN  20.0
1  13       NaN  24.0
2  14       NaN  31.0

 JOIN DATAFRAME
          name  age
id
12  Suryadev   20
13    Shakti   24

 MERGE DATAFRAME
    id      name  age
0  12  Suryadev   20
1  13    Shakti   24
```

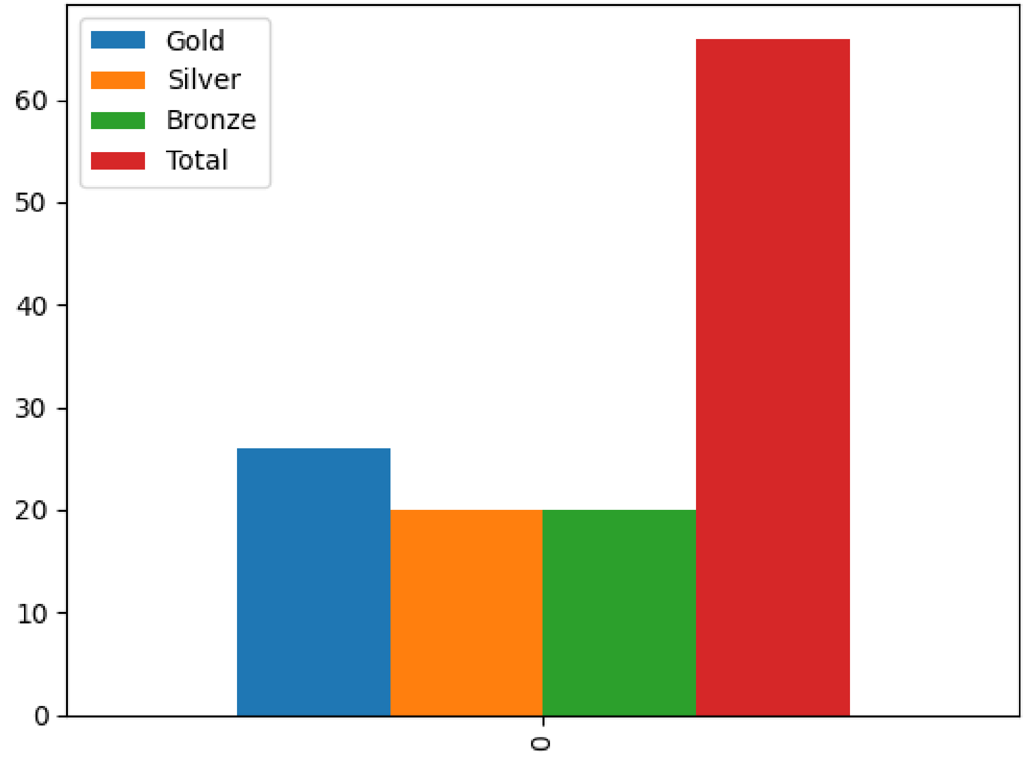
Q5. Write a program to draw bar graph for the following data for the Medal tally of Olympic games

Gold: 26 Silver: 20 Bronze: 20 Total: 66

[9]:

```
import matplotlib.pyplot as plt
data = {
 'Gold': [26],
 'Silver': [20],
 'Bronze': [20],
 'Total': [66]
}
df = pd.DataFrame(data)
print(df)
df.plot(kind='bar', width = 0.9)
plt.show()
```

```
   Gold  Silver  Bronze  Total
0    26      20      20     66
```

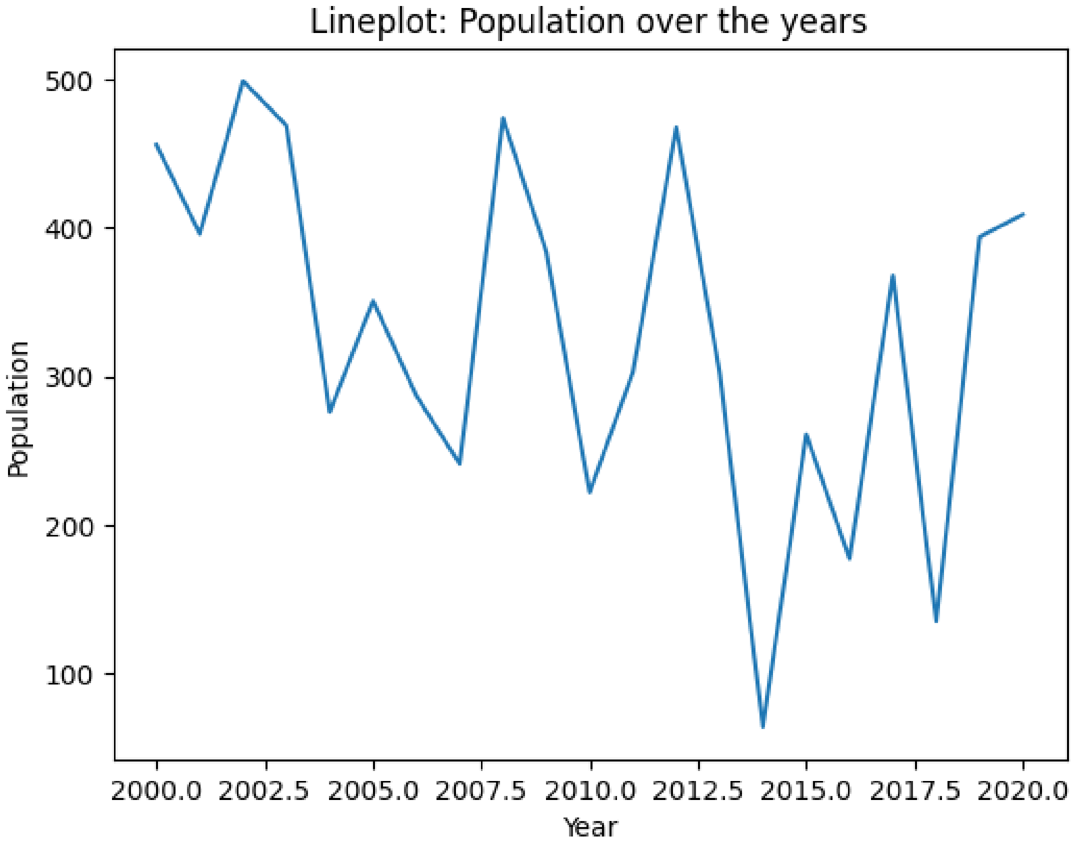


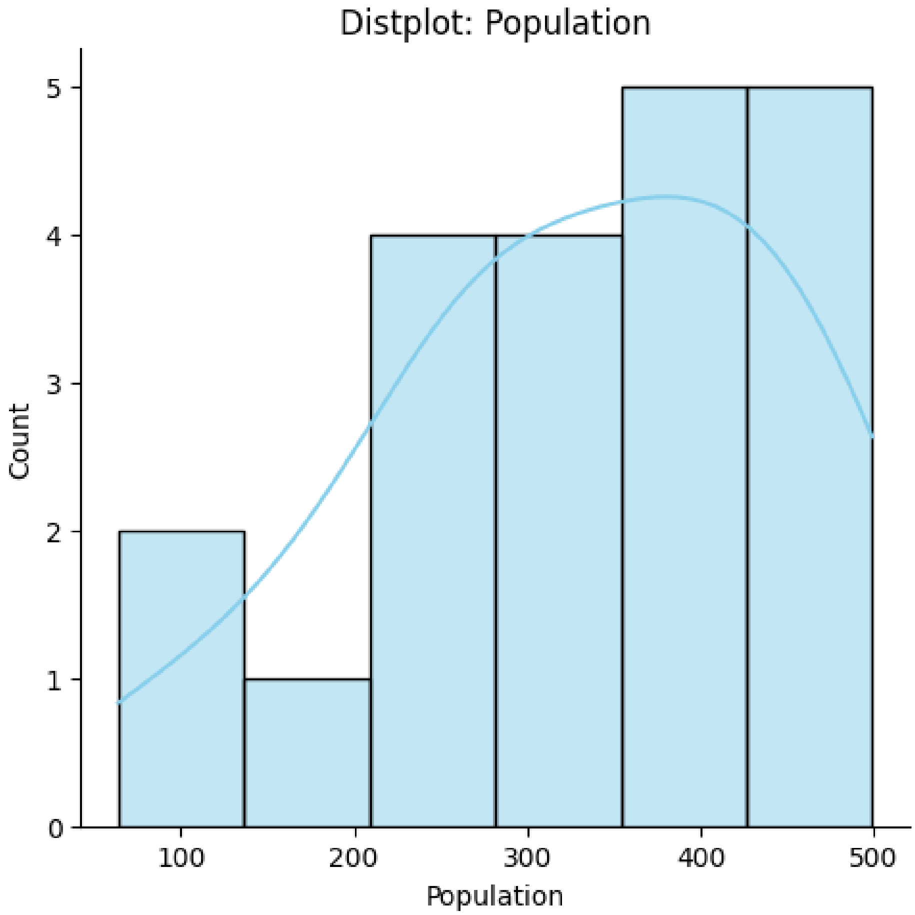
Q6. Implementing Line plot, Dist plot, Lmplot, Count plot using Seaborn library

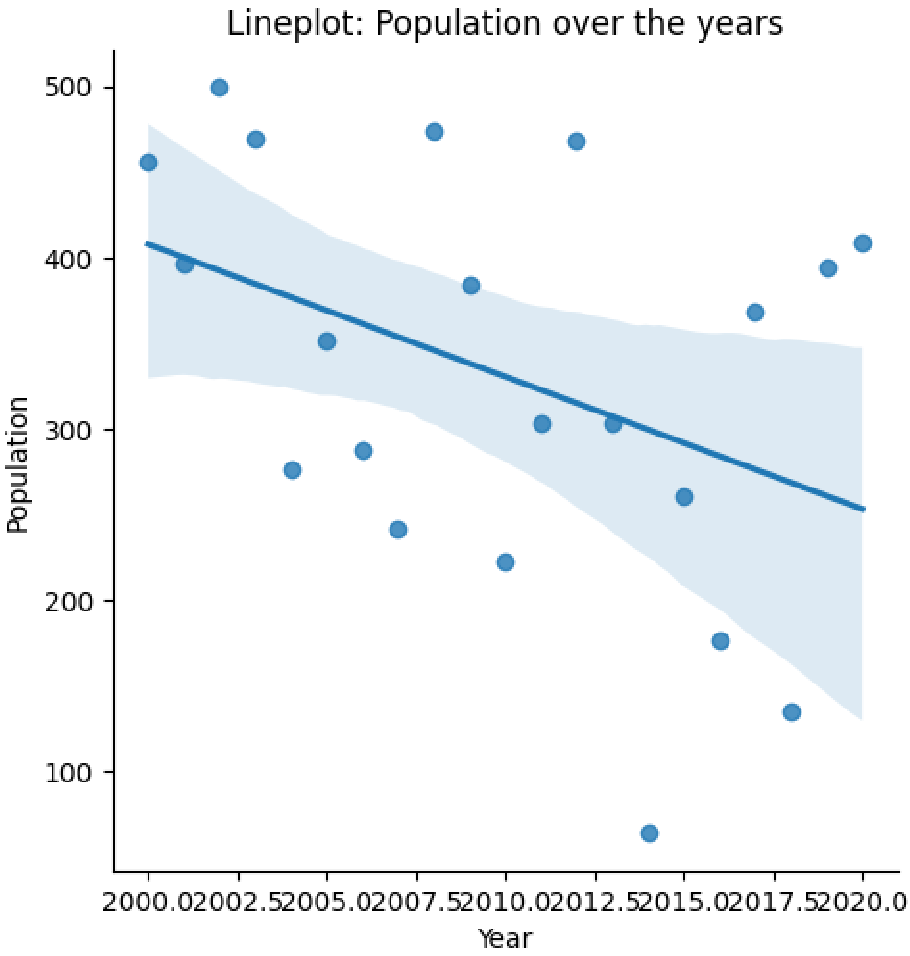
[13]:

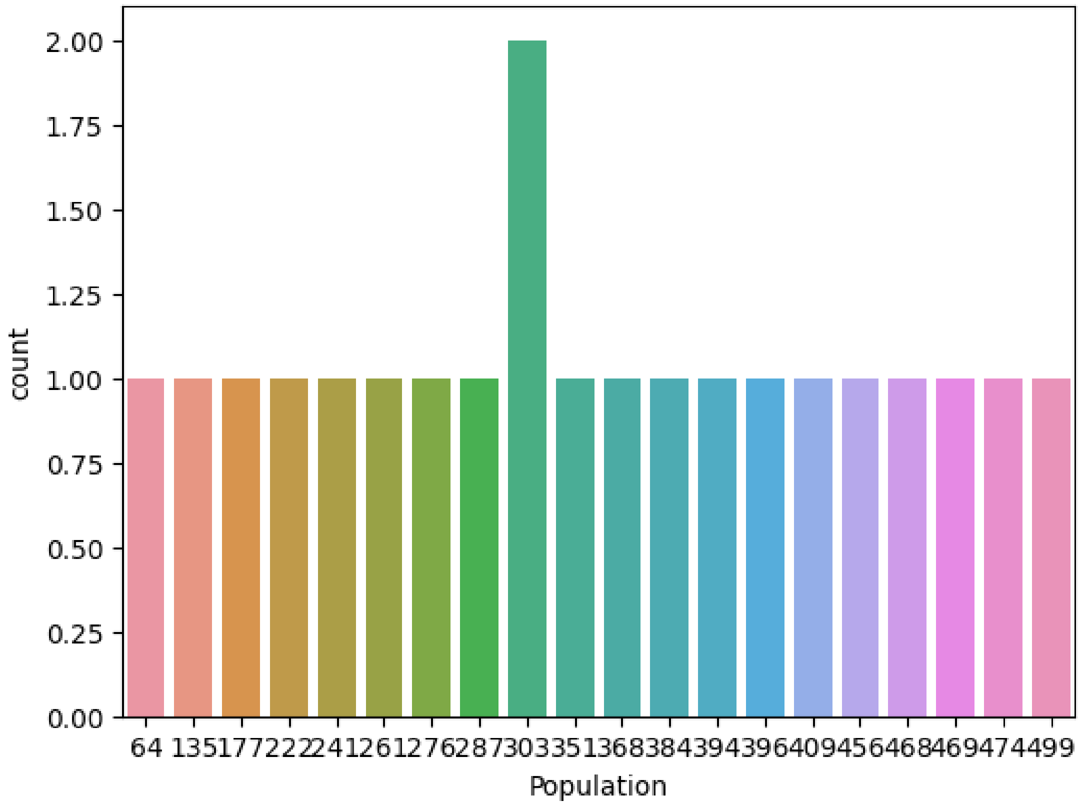
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
years = np.arange(2000, 2021, 1, dtype = int)
population = np.random.randint(50, 500, 21)
```

```
df = pd.DataFrame({'Year': years, 'Population': population })
sns.lineplot(x='Year', y='Population', data = df)
plt.title("Lineplot: Population over the years")
sns.displot(df['Population'], kde=True, color = 'skyblue')
plt.title("Distplot: Population")
plt.tight_layout()
plt.show()
sns.lmplot(x='Year', y='Population', data = df)
plt.title("Lineplot: Population over the years")
plt.show()
sns.countplot(x = 'Population', data = df)
plt.show()
```




Distplot: Population
Count
0
1
2
3
4
5
100
200
300
400
500
Population


Lineplot: Population over the years
500
400
300
200
100
Population
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
Year



Q7. Create a DataFrame namely aid that stores aid (Toys, books, uniform, shoes) by NGO's for different states. Write a program to display the aid for: - (a) Books and Uniforms only (b) Shoes only

[16]:

```
import pandas as pd
import numpy as np
data = {

 'State': ['Delhi','Haryana', 'Uttar Pradesh', 'Rajasthan', 'Gujarat',␣
 ↪'Kerala', 'Maharashtra', ],
 'Toys' : np.random.randint(5000, 8000, 7, dtype= int),
 'Books' : np.random.randint(10000, 15000, 7, dtype = int),
 'uniform': np.random.randint(5000, 8000, 7, dtype= int),
 'Shoes' : np.random.randint(5000, 8000, 7, dtype= int)
}
aid = pd.DataFrame(data)
print(aid)
print("\n(a) Aid for Books and Uniforms only:")
print(aid[['State', 'Books', 'uniform']])
print("\n(b) Aid for Shoes only:")
print(aid[['State', 'Shoes']])
```

```
            State  Toys  Books  uniform  Shoes
0           Delhi  5772  10153     5403   6235
1         Haryana  5604  14083     7450   5120
2  Uttar Pradesh  7126  12675     6381   7104
3       Rajasthan  6634  13703     6416   7039
4         Gujarat  6544  11517     7735   6002
5          Kerala  6892  14313     6564   7866
6     Maharashtra  6896  14854     7841   6165

(a) Aid for Books and Uniforms only:
            State  Books  uniform
0           Delhi  10153     5403
1         Haryana  14083     7450
2  Uttar Pradesh  12675     6381
3       Rajasthan  13703     6416
4         Gujarat  11517     7735
5          Kerala  14313     6564
6     Maharashtra  14854     7841

(b) Aid for Shoes only:
            State  Shoes
0           Delhi   6235
1         Haryana   5120
2  Uttar Pradesh   7104
3       Rajasthan   7039
4         Gujarat   6002
5          Kerala   7866
6     Maharashtra   6165
```

Q8. Create a DataFrame df having Name, Gender, Position, City, Age, Projects. Write a program to summarize how many projects are being handled by each position for each city? Use pivot()

[19]:
```python
import pandas as pd
data = {
 'Name': ['Sourav', 'Aarti', 'Suryadev', 'Viswam', 'Wusat'],
 'Gender': ['Male', 'Female', 'Male', 'Male', 'Female'],
 'Position': ['Analyst', 'Developer', 'Analyst', 'Developer', 'Manager'],
 'City': ['Boston', 'Los Angeles', 'Boston', 'Los Angeles', 'Boston'],
 'Age': [20, 25, 35, 27, 32],
 'Projects': [5, 3, 6, 4, 7]
}
df = pd.DataFrame(data)
grouped_df = df.groupby(['Position', 'City'], as_index=False)['Projects'].sum()
pt = grouped_df.pivot(index='Position', columns='City', values = 'Projects').
 ↪fillna(0)
print(pt)
```

```
City       Boston  Los Angeles
Position
Analyst      11.0          0.0
Developer     0.0          7.0
Manager       7.0          0.0
```

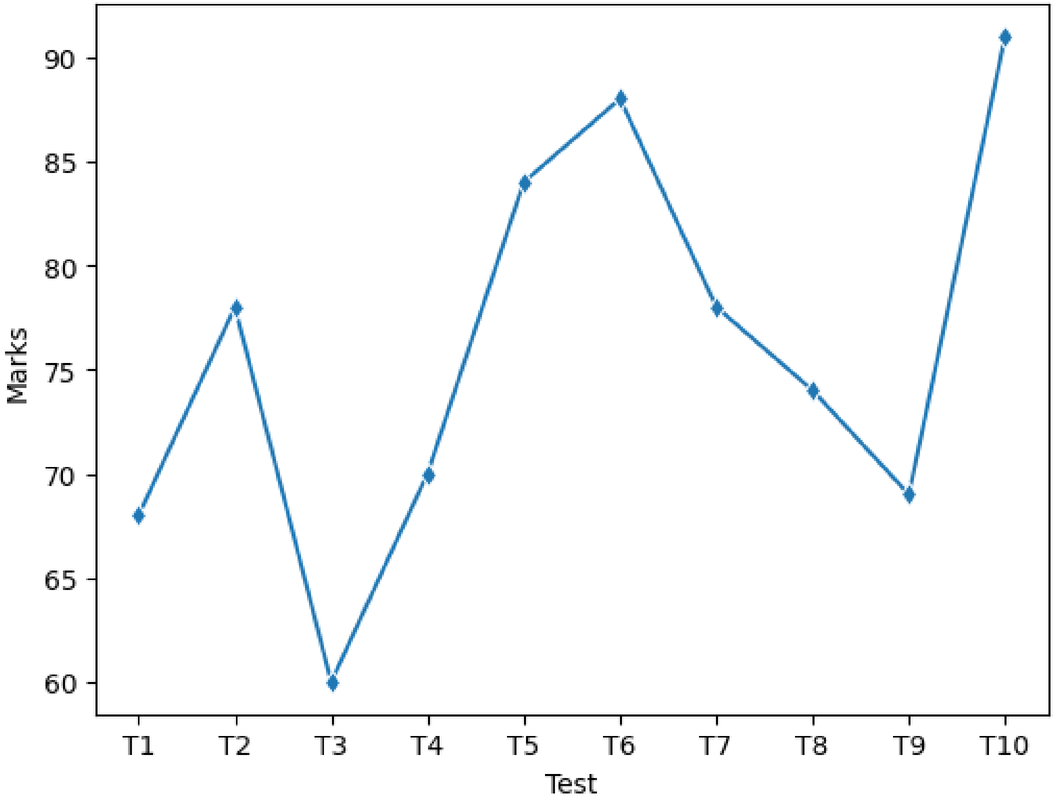
Q9. Marks is a list that stores marks of a student in 10-unit test. Write a program to plot Line chart for the student's performance in these 10 tests

[21]:

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
Test = ['T1', 'T2', 'T3', 'T4', 'T5', 'T6', 'T7', 'T8', 'T9', 'T10']
marks = np.random.randint(30, 100, 10)
df = pd.DataFrame({'Test': Test, 'Marks': marks})
sns.lineplot(data=df, x='Test', y='Marks', marker = 'd')
```

[21]:

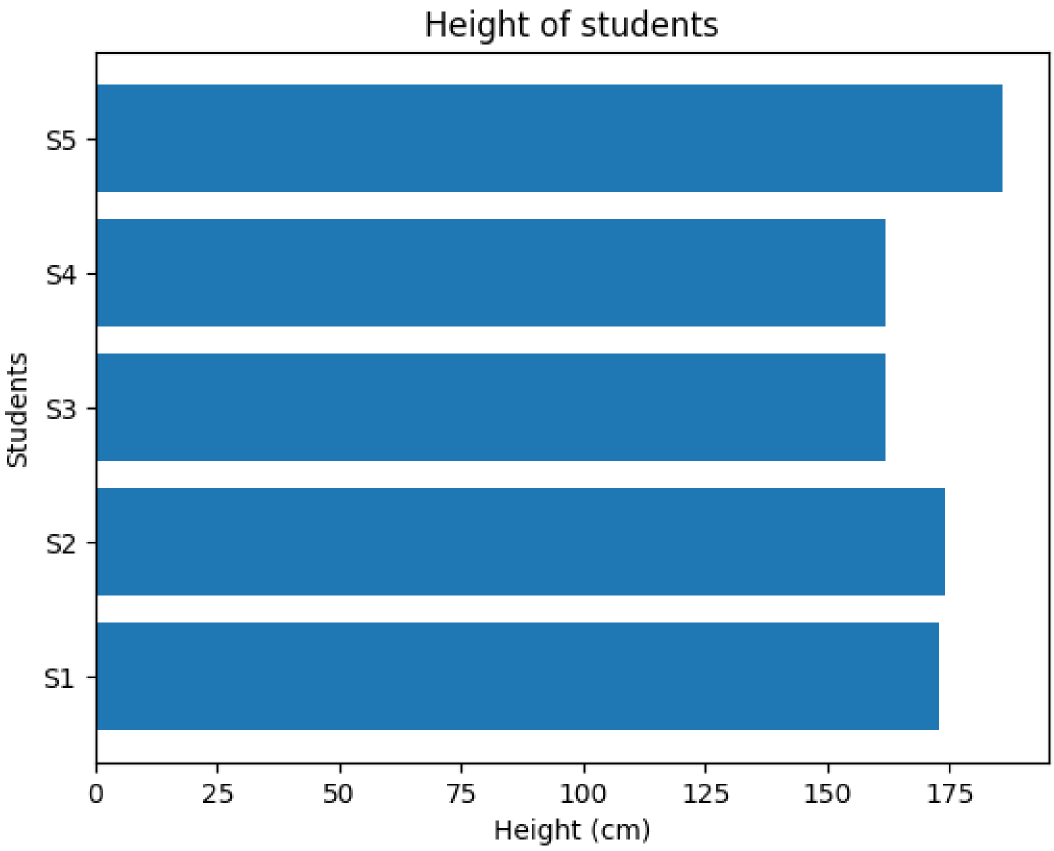
```
<Axes: xlabel='Test', ylabel='Marks'>
```



Q10. Write a program to plot a horizontal bar chart from the height of some students

[22]:

```
import matplotlib.pyplot as plt
import numpy as np
students = ['S1', 'S2', 'S3', 'S4', 'S5']
heights = np.random.randint(150, 190, len(students))
# Create a horizontal bar chart
plt.barh(students, heights)
plt.xlabel('Height (cm)')
plt.ylabel('Students')
plt.title('Height of students')
plt.show()
```



Q11. Write a program to implement Covariance.

[24]:

```
import numpy as np
d1 = np.array([11,12,13,14,15])
d2 = np.array([30,40,50,10,20])
def covariance(X, Y):
 n = len(X)
 mean_X = np.mean(X)
 mean_Y = np.mean(Y)
 cov = np.sum((X - mean_X) * (Y - mean_Y)) / (n - 1)
 return cov
covariance_value = covariance(d1, d2)
print(f"Covariance between X and Y: {covariance_value}")
print("\n Covariance through np.cov()\n", np.cov(d1, d2))
```

```
Covariance between X and Y: -12.5

 Covariance through np.cov()
 [[  2.5 -12.5]
 [-12.5 250. ]]
```

Q12. Segmentation: Clustering (K-Means) Customer Personality Analysis is a detailed analysis of a company's ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers. Customer personality analysis helps a business to modify its product based on its target customers from different types of customer segments. For example, instead of spending money to market a new product to every customer in the company's database, a company can analyze which customer segment is most likely to buy the product and then market the product only on that particular segment. link to dataset: https://www.kaggle.com/datasets/imakash3011/customerpersonality-analysis Refer https://github.com/ibrahim-ogunbiyi/CustomerSegmentation/blob/main/Customer%20Segmentat

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

df = pd.read_csv("C:\SOURAV\BCA\Semester-6\Data Visualisation and␣
 ↪Analytics\Assignments\Supermarket store Dataset.csv")
df.head()
```

[1]:
```
   Store ID   Store_Area  Items_Available  Daily_Customer_Count  Store_Sales
0          1        1659             1961                   530        66490
1          2        1461             1752                   210        39820
2          3        1340             1609                   720        54010
3          4        1451             1748                   620        53730
4          5        1770             2111                   450        46620
```

[41]:
```
df.describe()
```

[41]:
```
         Store ID   Store_Area  Items_Available  Daily_Customer_Count  \
count  896.000000   896.000000       896.000000            896.000000
mean   448.500000  1485.409598      1782.035714            786.350446
std    258.797218   250.237011       299.872053            265.389281
min      1.000000   775.000000       932.000000             10.000000
25%    224.750000  1316.750000      1575.500000            600.000000
```

```
50%    448.500000  1477.000000      1773.500000             780.000000
75%    672.250000  1653.500000      1982.750000             970.000000
max    896.000000  2229.000000      2667.000000            1560.000000

         Store_Sales
count     896.000000
mean    59351.305804
std     17190.741895
min     14920.000000
25%     46530.000000
50%     58605.000000
75%     71872.500000
max    116320.000000
```

[42]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 896 entries, 0 to 895
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Store ID              896 non-null    int64
 1   Store_Area            896 non-null    int64
 2   Items_Available       896 non-null    int64
 3   Daily_Customer_Count  896 non-null    int64
 4   Store_Sales           896 non-null    int64
dtypes: int64(5)
memory usage: 35.1 KB
```

[43]:

```python
df.isnull().sum()
```

[43]:

```
Store ID                0
Store_Area              0
Items_Available         0
Daily_Customer_Count    0
Store_Sales             0
dtype: int64
```

[47]:

```python
df.duplicated().sum()
```

[47]:

```
0
```

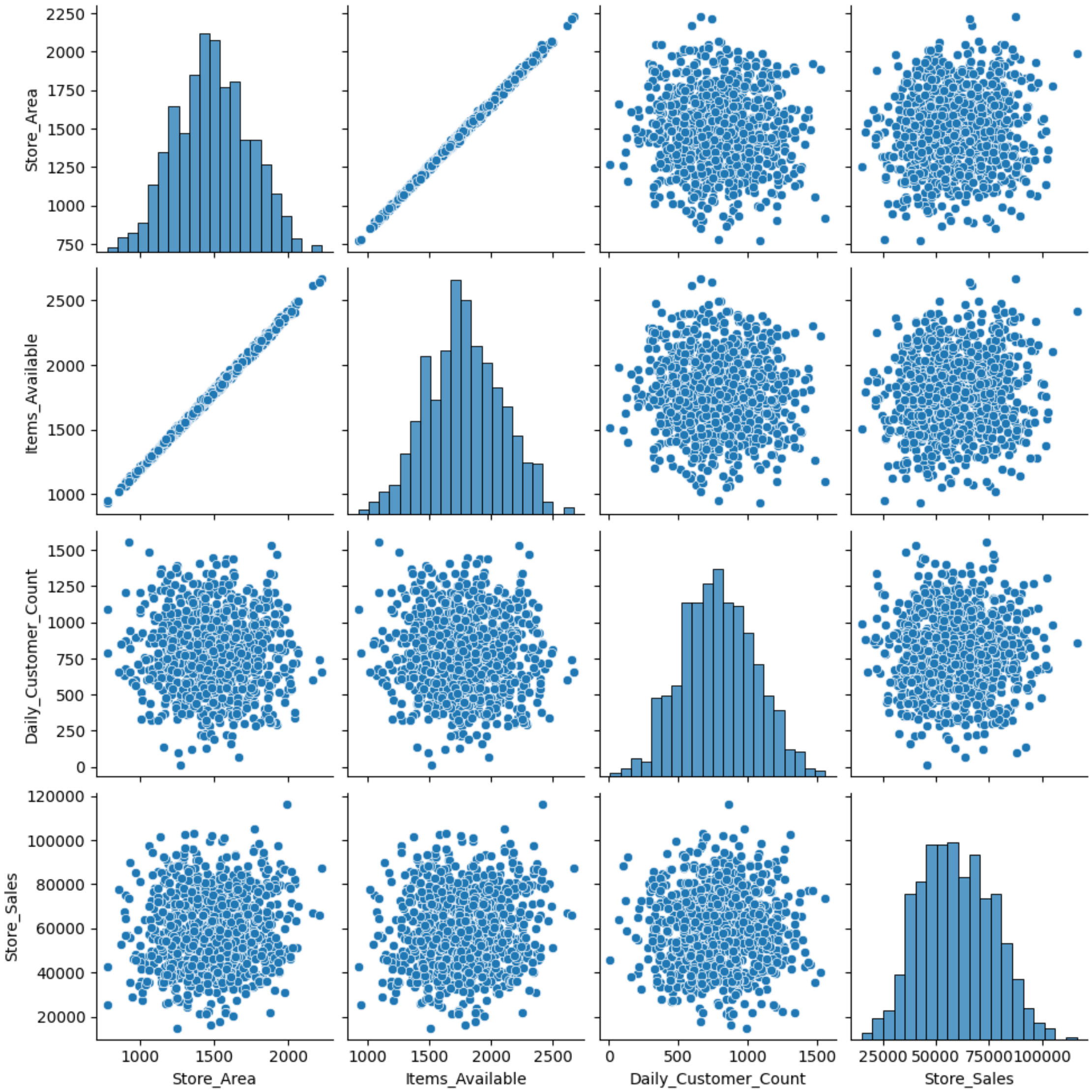
[ ]:

```python
df.drop(['Store ID'], axis=1)
```

[51]:

```python
plt.figure(figsize=(3,3))
sns.pairplot(data = df)
plt.show()
```

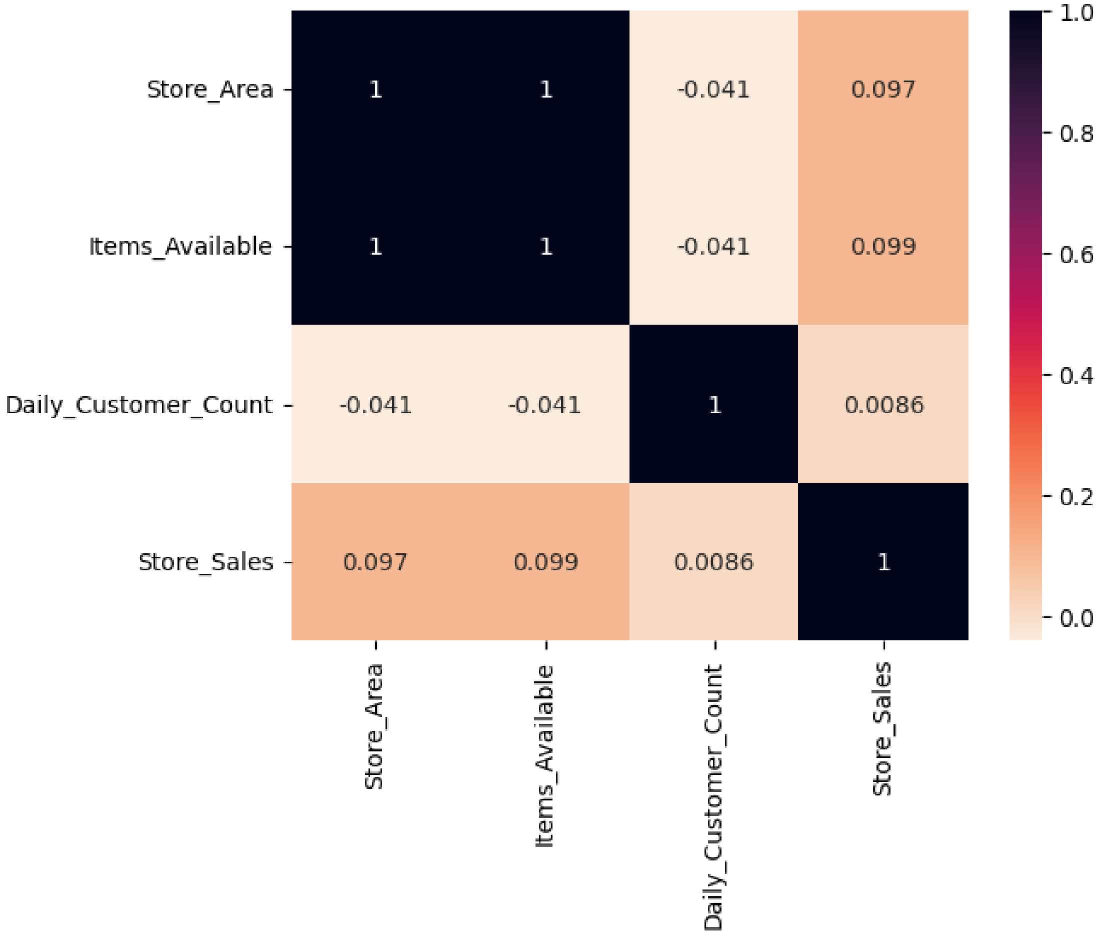
```
<Figure size 300x300 with 0 Axes>
```



The distribution of Store_Area, Items_Available, Daily_Customer_Count, Store_Sales is normal.Item_Available and Store_Area has high positive correlation

```
[52]: sns.heatmap(df.corr(),annot=True, cmap= 'rocket_r')
```

```
[52]: <Axes: >
```



From above heatmap, we can notice that Store_Area and Items_Available has a linear relationship Therefore, it will be a good idea to drop one of them. And has more impact on Store_Sales

[53]:

```
df = df.drop(['Items_Available'], axis=1)
```

We'll only use 'Store_Area' and 'Store_Sales' columns to get insights into store characteristics and sales performance

[56]:

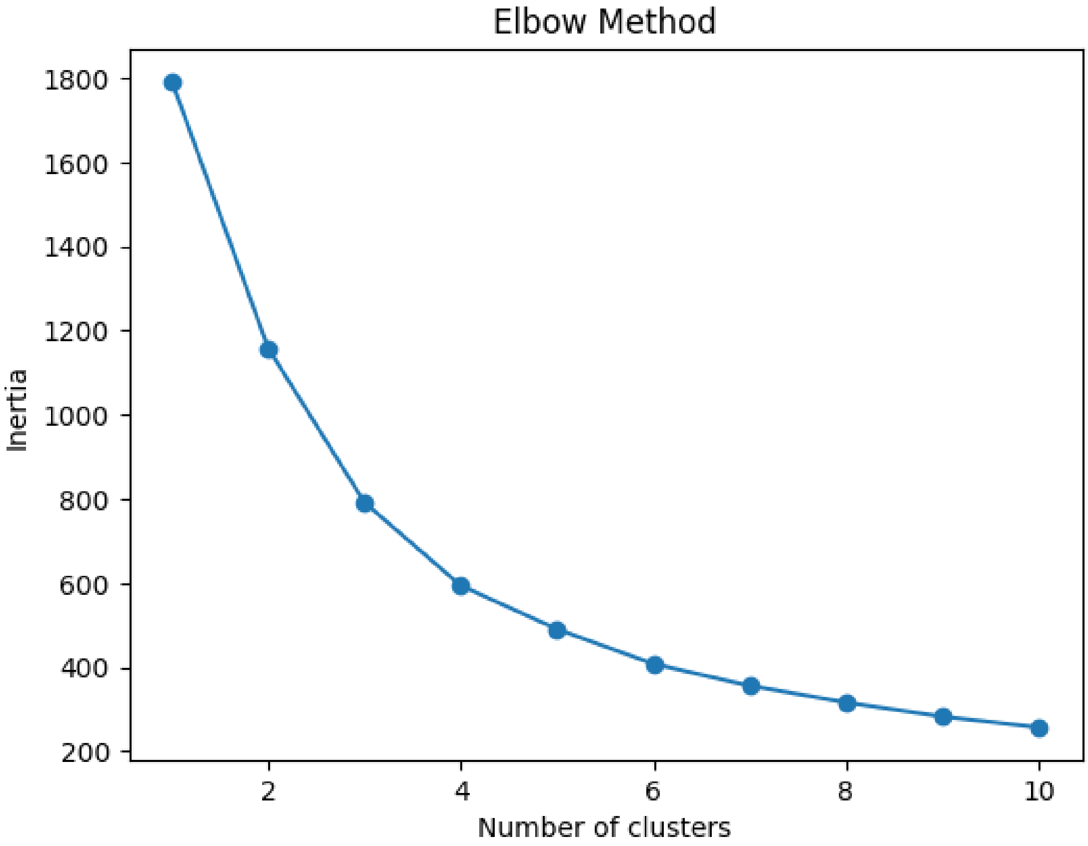
```
df = df[['Store_Area', 'Store_Sales']]
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df)
```

[60]:

```
from sklearn.cluster import KMeans

inertia = []
for i in range(1, 11):
    kmeans = KMeans (n_clusters=i, n_init=10, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```

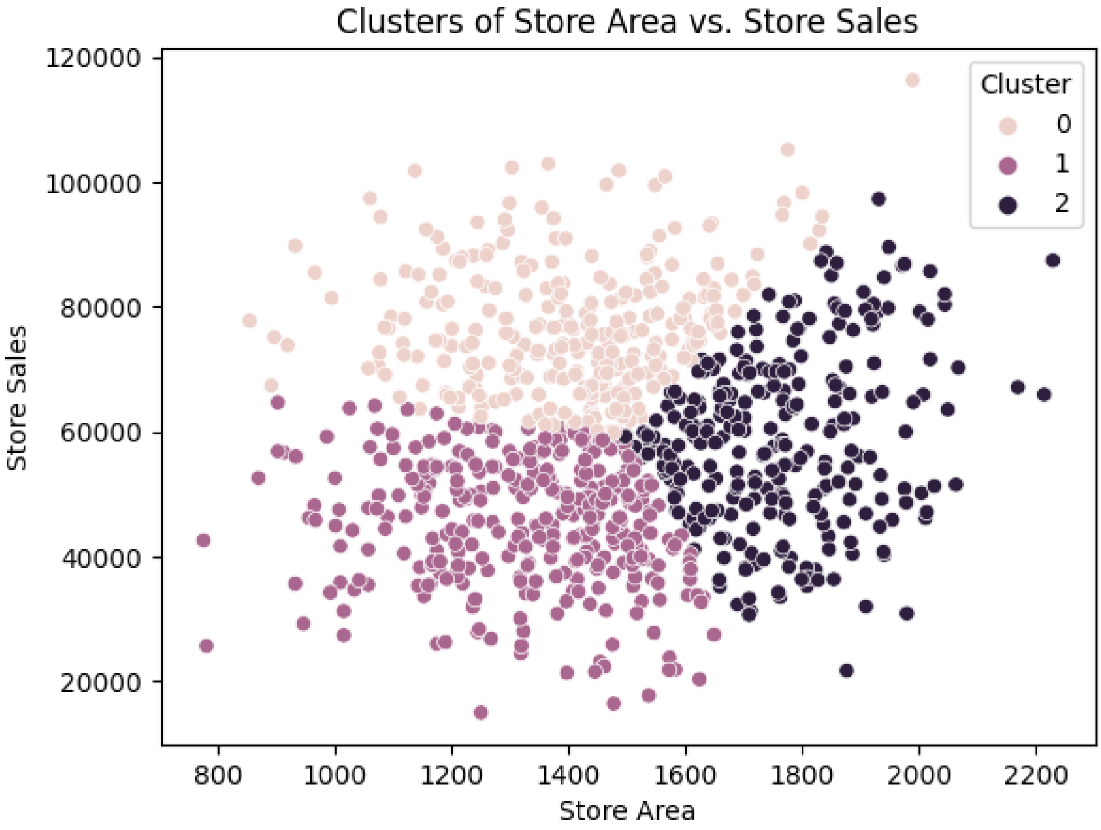


[59]:
```
num_clusters = 3
#Perform K-means clustering
kmeans = KMeans(n_clusters=num_clusters, n_init=10, random_state=42)
kmeans.fit(scaled_features)

# Add cluster Labels to the DataFrame
df['Cluster'] = kmeans.labels_

# Visualize the clusters
```

```
sns.scatterplot(x= 'Store_Area', y='Store_Sales', hue='Cluster', data=df)
plt.xlabel('Store Area')
plt.ylabel('Store Sales')
plt.title('Clusters of Store Area vs. Store Sales')
plt.show()
```



[ ]: