```
A) Conduct simple data exploration and data analysis on the given dataset .

(**number your questions/tasks completed accordingly using python comments in each cell
of your code)

Applied Data Science Exam Project

1) Display the number of attributes available in the dataset (exam_dataset.csv)?
2) Find the dimension number of this dataset
3) Display the average of these attributes: ‘Age’, ‘Bonus’ and ‘Years at
Company’. Be sure to round your answer to 4 decimal places
4) Find the minimum and maximum ‘Bonus’
5) What are the departments in this dataset?
6) Provide graphical visualization by plotting Histogram of ‘JobSatisfaction’ vs staff
numbers’
7) Find the correlation between ‘Bonus’ and ‘JobSatisfaction’
8) Based on your findings, discuss briefly(you can either use comments/markdown to
write your answer) on:
a) Range of bonus at Company A
b) Most and Least Frequent JobSatisfaction at Company A
c) Discuss your observation on the distribution of bonus values
d) Is there a linear relationship between bonus and JobSatisfaction
at the company?
```

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

In [2]:
```
data = pd.read_csv(r"C:\Users\meorh\Desktop\Exam\exam_dataset.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	Age	BusinessTravel	MonthlyIncome	JobSatisfaction	Bonus	Department	DistanceFromHome	Education	EducationField	EmployeeCount	...	J
0	41	Travel_Rarely	5993	4	17979	Sales	1	2	Life Sciences	1	...	E
1	49	Travel_Frequently	5130	2	20520	Research & Development	8	1	Life Sciences	1	...	Re S
2	37	Travel_Rarely	2090	3	6270	Research & Development	2	2	Other	1	...	Lal Tec
3	33	Travel_Frequently	2909	3	8727	Research & Development	3	4	Life Sciences	1	...	Re S
4	27	Travel_Rarely	3468	2	10404	Research & Development	2	1	Medical	1	...	Lal Tec

5 rows × 24 columns

In [4]:
```
#1) Display the number of attributes available in the dataset (exam_dataset.csv)?

data.info()

print(" ")
print("Number of attributes:", len(data.columns))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Age                      1470 non-null   int64 
 1   BusinessTravel           1470 non-null   object
 2   MonthlyIncome            1470 non-null   int64 
 3   JobSatisfaction          1470 non-null   int64 
 4   Bonus                    1470 non-null   int64 
 5   Department               1470 non-null   object
 6   DistanceFromHome         1470 non-null   int64 
 7   Education                1470 non-null   int64 
 8   EducationField           1470 non-null   object
 9   EmployeeCount            1470 non-null   int64 
 10  EmployeeNumber           1470 non-null   int64 
 11  EnvironmentSatisfaction  1470 non-null   int64 
 12  Gender                   1470 non-null   object
 13  JobLevel                 1470 non-null   int64 
 14  JobRole                  1470 non-null   object
 15  MaritalStatus            1470 non-null   object
 16  PerformanceRating        1470 non-null   int64 
 17  StockOptionLevel         1470 non-null   int64 
 18  TrainingTimesLastYear    1470 non-null   int64 
 19  WorkLifeBalance          1470 non-null   int64 
 20  YearsAtCompany           1470 non-null   int64 
 21  YearsSinceLastPromotion  1470 non-null   int64 
 22  OverTime                 1470 non-null   object
 23  Attrition                1470 non-null   object
dtypes: int64(16), object(8)
memory usage: 275.8+ KB

Number of attributes: 24
```

In [5]:
```
#2) Find the dimension number of this dataset

print("Number of dimension number:", data.ndim)
```

```
Number of dimension number: 2
```

In [6]:
```
#3) Display the average of these attributes: ‘Age’, ‘Bonus’ and ‘Years at Company’. Be sure to round your answer to

avg_age = np.mean(data['Age'])
avg_bonus = np.mean(data['Bonus'])
avg_years_at_company = np.mean(data['YearsAtCompany'])

print("Average of 'Age':", round(avg_age, 4))
print("Average of 'Bonus':", round(avg_bonus, 4))
print("Average of 'Years at company':", round(avg_years_at_company, 4))
```

```
Average of 'Age': 36.9238
Average of 'Bonus': 20479.5014
Average of 'Years at company': 7.0082
```

In [7]:
```
#4) Find the minimum and maximum ‘Bonus’

min_bonus = np.min(data['Bonus'])
max_bonus = np.max(data['Bonus'])

print("Minimum of 'Bonus':", min_bonus)
print("Maximum of 'Bonus':", max_bonus)
```

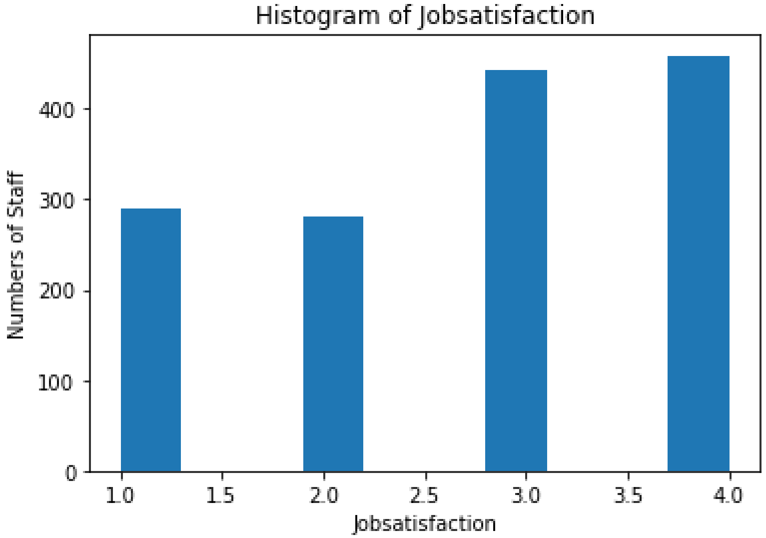
```
Minimum of 'Bonus': 3027
Maximum of 'Bonus': 79892
```

In [8]:
```
#5) What are the departments in this dataset?

print("Department in this dataset", data['Department'].unique())
```

```
Department in this dataset ['Sales' 'Research & Development' 'Human Resources']
```

In [9]:

```
#6) Provide graphical visualization by plotting Histogram of ‘JobSatisfaction’ vs staff numbers’

plt.hist(data['JobSatisfaction'])
plt.title("Histogram of Jobsatisfaction")
plt.xlabel("Jobsatisfaction")
plt.ylabel("Numbers of Staff")
plt.show()
```



In [10]:

```
#7) Find the correlation between ‘Bonus’ and ‘JobSatisfaction’

value = data.iloc[:,[3,4]]
corr = value.corr()
print(corr)
```

```
                 JobSatisfaction     Bonus
JobSatisfaction         1.000000 -0.003652
Bonus                  -0.003652  1.000000
```

8) Based on your findings, discuss briefly(you can either use comments/markdown to write your answer) on:

a) Range of bonus at Company A
b) Most and Least Frequent JobSatisfaction at Company A
c) Discuss your observation on the distribution of bonus values
d) Is there a linear relationship between bonus and JobSatisfaction
at the company?

In [11]:
```python
#a) Range of bonus at Company A

range_bonus = max_bonus-min_bonus
print("Range of bonus at Company A:", range_bonus)
```

```
Range of bonus at Company A: 76865
```

In [12]:
```python
#b) Most and Least Frequent JobSatisfaction at Company A

data['JobSatisfaction'].describe()
```

Out[12]:
```
count    1470.000000
mean        2.728571
std         1.102846
min         1.000000
25%         2.000000
50%         3.000000
75%         4.000000
max         4.000000
Name: JobSatisfaction, dtype: float64
```

In [13]:
```
#b) Most and Least Frequent JobSatisfaction at Company A
#finding most frequent and least through sort values

data['JobSatisfaction'].sort_values()
```

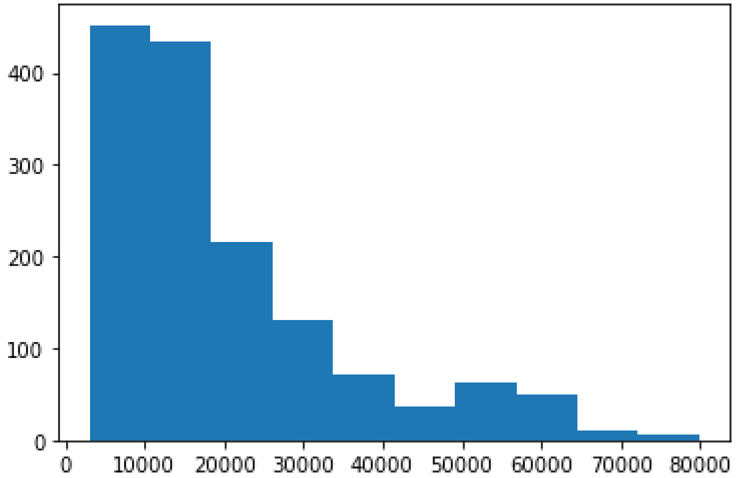
Out[13]:
```
734     1
1077    1
1078    1
427     1
1083    1
       ..
1036    4
488     4
489     4
454     4
0       4
Name: JobSatisfaction, Length: 1470, dtype: int64
```

In [14]:
```
#b) Most and Least Frequent JobSatisfaction at Company A

print("Most frequent Jobsatisfaction: 4")

print("Least frequent Jobsatisfaction: 1")
```

```
Most frequent Jobsatisfaction: 4
Least frequent Jobsatisfaction: 1
```

In [15]:
```
#c) Discuss your observation on the distribution of bonus values

plt.hist(data['Bonus'])
data['Bonus'].describe()
```

Out[15]:
```
count     1470.000000
mean     20479.501361
std      15066.272964
min       3027.000000
25%       9333.750000
50%      15484.500000
75%      26103.750000
max      79892.000000
Name: Bonus, dtype: float64
```



#c) Discuss your observation on the distribution of bonus values

Based on the histogram plotted, distribution of bonus value is towards right.

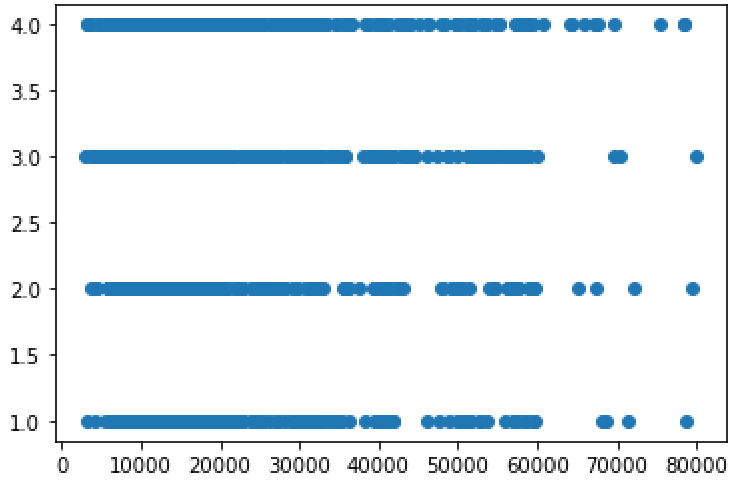
Also, we can see most of people in Company A receive an average of mean of 20479.501361, minimum of 3027.000000, and maximum value of 79892.000000.

In [16]:
```
#d) Is there a linear relationship between bonus and JobSatisfaction at the company?

value = data.iloc[:,[3,4]]
corr = value.corr()
print(corr)
```

```
                 JobSatisfaction     Bonus
JobSatisfaction         1.000000 -0.003652
Bonus                  -0.003652  1.000000
```

In [17]:
```
plt.scatter(data['Bonus'], data['JobSatisfaction'])
```

Out[17]: <matplotlib.collections.PathCollection at 0x1ecf36fac70>




```
...
#d) Is there a linear relationship between bonus and JobSatisfaction at the company?

Based on the correlation value of -0.003652, we can see it is approaching 0 value indicating it has very low correlation between both value.
...
```

```
...
B) Classification using Random Forest for: ‘Age’, ‘BusinessTravel’, ‘MonthlyIncome’
and ‘JobSatisfaction’ to predict ‘Attrition’. Sample steps are as below:
1) Import necessary libraries
2) Import dataset (exam_dataset.csv)
3) Allocate the relevant attributes as input and output
4) Use LabelEncoder to encode categorical data
5) Split your data into training and test sets with the appropriate proportions
6) Normalized your data using StandardScaler
7) Fit the and predict results using the Classifier
8) Evaluate your results using confusion matrix and calculate the prediction accuracy
9) Discuss your results and findings
...
```

In [18]:
```
#1) Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [19]:
```
#2) Import dataset (exam_dataset.csv)

data = pd.read_csv(r"C:\Users\meorh\Desktop\Exam\exam_dataset.csv")
data.head()
```

Out[19]:

	Age	BusinessTravel	MonthlyIncome	JobSatisfaction	Bonus	Department	DistanceFromHome	Education	EducationField	EmployeeCount	...	J
0	41	Travel_Rarely	5993	4	17979	Sales	1	2	Life Sciences	1	...	E
1	49	Travel_Frequently	5130	2	20520	Research & Development	8	1	Life Sciences	1	...	Re S
2	37	Travel_Rarely	2090	3	6270	Research & Development	2	2	Other	1	...	Lal Tec
3	33	Travel_Frequently	2909	3	8727	Research & Development	3	4	Life Sciences	1	...	Re S
4	27	Travel_Rarely	3468	2	10404	Research & Development	2	1	Medical	1	...	Lal Tec

5 rows × 24 columns

In [20]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Age                      1470 non-null   int64 
 1   BusinessTravel           1470 non-null   object
 2   MonthlyIncome            1470 non-null   int64 
 3   JobSatisfaction          1470 non-null   int64 
 4   Bonus                    1470 non-null   int64 
 5   Department               1470 non-null   object
 6   DistanceFromHome         1470 non-null   int64 
 7   Education                1470 non-null   int64 
 8   EducationField           1470 non-null   object
 9   EmployeeCount            1470 non-null   int64 
 10  EmployeeNumber           1470 non-null   int64 
 11  EnvironmentSatisfaction  1470 non-null   int64 
 12  Gender                   1470 non-null   object
 13  JobLevel                 1470 non-null   int64 
 14  JobRole                  1470 non-null   object
 15  MaritalStatus            1470 non-null   object
 16  PerformanceRating        1470 non-null   int64 
 17  StockOptionLevel         1470 non-null   int64 
 18  TrainingTimesLastYear    1470 non-null   int64 
 19  WorkLifeBalance          1470 non-null   int64 
 20  YearsAtCompany           1470 non-null   int64 
 21  YearsSinceLastPromotion  1470 non-null   int64 
 22  OverTime                 1470 non-null   object
 23  Attrition                1470 non-null   object
dtypes: int64(16), object(8)
memory usage: 275.8+ KB
```

In [21]:
```
#3) Allocate the relevant attributes as input and output

x = data.iloc[:,[0,1,2,3]].values
y = data.iloc[:,23].values

x,y
```

Out[21]:
```
(array([[41, 'Travel_Rarely', 5993, 4],
        [49, 'Travel_Frequently', 5130, 2],
        [37, 'Travel_Rarely', 2090, 3],
        ...,
        [27, 'Travel_Rarely', 6142, 2],
        [49, 'Travel_Frequently', 5390, 2],
        [34, 'Travel_Rarely', 4404, 3]], dtype=object),
 array(['Yes', 'No', 'Yes', ..., 'No', 'No', 'No'], dtype=object))
```

In [22]:
```
#4) Use LabelEncoder to encode categorical data

from sklearn.preprocessing import LabelEncoder

labelencoder_x = LabelEncoder()
x[:,1] = labelencoder_x.fit_transform(x[:,1])

labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

In [23]:
```
x,y
```

Out[23]:
```
(array([[41, 2, 5993, 4],
        [49, 1, 5130, 2],
        [37, 2, 2090, 3],
        ...,
        [27, 2, 6142, 2],
        [49, 1, 5390, 2],
        [34, 2, 4404, 3]], dtype=object),
 array([1, 0, 1, ..., 0, 0, 0]))
```

In [24]:
```
#5) Split your data into training and test sets with the appropriate proportions

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=0)
```

In [25]:
```
#6) Normalized your data using StandardScaler

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

In [26]:
```
#7) Fit the and predict results using the Classifier

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=50, criterion='entropy', random_state=0)
classifier.fit(x_train, y_train)

prediction = classifier.predict(x_test)
```

In [27]:

```
y_test, prediction
```

Out[27]:

```
(array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
        1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
        0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 1, 0, 0, 0, 1, 0, 0]),
 array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0]))
```

In [28]:
```
#8) Evaluate your results using confusion matrix and calculate the prediction accuracy

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, prediction)
print(cm)
```

```
[[233  12]
 [ 42   7]]
```

In [29]:
```
#8) Evaluate your results using confusion matrix and calculate the prediction accuracy

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, prediction)
print(accuracy)
```

```
0.8163265306122449
```

In [30]:
```
#9) Discuss your results and findings

from sklearn.metrics import classification_report

report = classification_report(y_test, prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.85      0.95      0.90       245
           1       0.37      0.14      0.21        49

    accuracy                           0.82       294
   macro avg       0.61      0.55      0.55       294
weighted avg       0.77      0.82      0.78       294
```

Based on the accuracy value of 0.8163265306122449, column ‘Age’, ‘BusinessTravel’, ‘MonthlyIncome’ and ‘JobSatisfaction’ is quite good to measure column 'Attrition value'.

C)Clustering comparison between K-Means and DBSCAN
1) Perform K-Means clustering (use WCSS to help find best K value) on the given dataset, display clustering results with graphical visualization, provide any necessary comments and discussions.
2) Perform DBSCAN clustering (use knee locator to help find optimal parameter) on the given dataset, display clustering results with graphical visualization, provide any necessary comments and discussions.
3) Conduct comparison studies on the two techniques (K-Means and DBSCAN), with graphical visualization comparisons, discuss your results and decide on whether:
a. K-Means is the better clustering technique for this dataset or,
b. DBSCAN is the better clustering technique for this dataset, or
c. There’s no clear distinction between the two techniques for this dataset

In [31]:
```
#1) Perform K-Means clustering (use WCSS to help find best K value) on the given dataset, display clustering results

import pandas as pd
import numpy as np
import matplotlib.pyplot
import seaborn as sns
```

In [32]:
```
#importing dataset
data = pd.read_csv(r"C:\Users\meorh\Desktop\Exam\clustering.csv")
data.head()
```

Out[32]:

	Unnamed: 0	A	B
0	0	0.329241	0.841783
1	1	1.697407	-0.236075
2	2	-0.831460	0.584743
3	3	1.825271	-0.297894
4	4	1.236577	0.121528

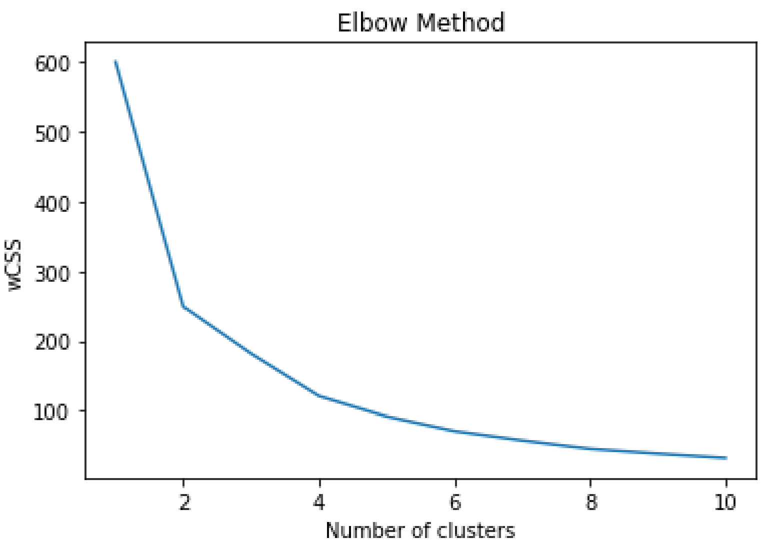
In [33]:
```
#allocating input
x = data.iloc[:,[1,2]].values
```

In [34]:
```
#scaling data
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
x_scaled
```

```
       [-0.55262164, -0.25998898],
       [-1.50496388,  0.62031921],
       [ 1.10513637, -1.01082203],
       [ 0.38189332, -1.64868256],
       [-0.57425797,  1.48875469],
       [ 1.69212349, -0.13186847],
       [ 0.64464852, -0.2842848 ],
       [-0.17188414, -1.00013217],
       [-0.40308758,  1.37018963],
       [ 0.47195018,  0.6674501 ],
       [-0.04455545,  1.28442231],
       [ 1.00641662, -1.62400292],
       [ 0.41258526,  0.85328116],
       [ 1.00260241, -1.63401717],
       [-1.77578325,  0.02797682],
       [ 0.14046294, -1.43902102],
       [-0.77072388, -0.06999854],
       [-1.60386009,  0.19275344],
       [ 0.06792247, -1.13251967],
       [ 0.15808104,  0.99907498],
```

In [35]:
```
#finding elbow method

from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state=0)
    kmeans.fit(x_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11), wcss)
plt.title("Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("wCSS")
plt.show()
```

```
C:\Users\meorh\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting th
e environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

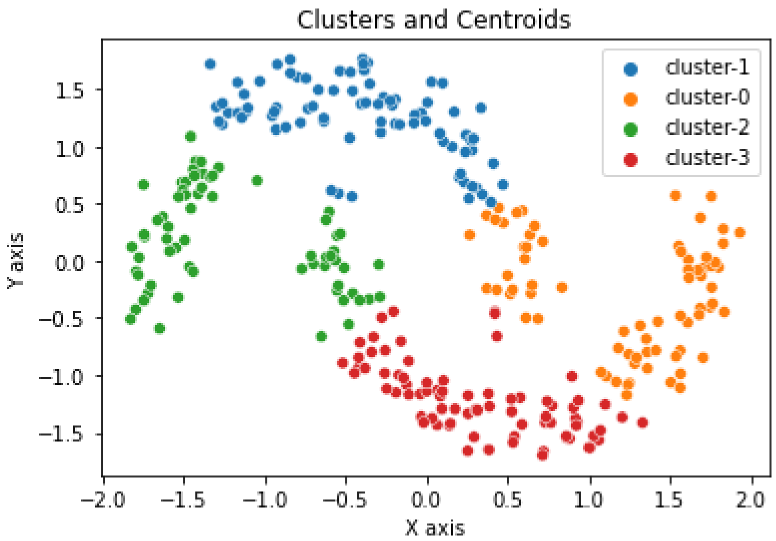


In [36]:
```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state=0)
y_kmeans = kmeans.fit_predict(x_scaled)
y_kmeans
```

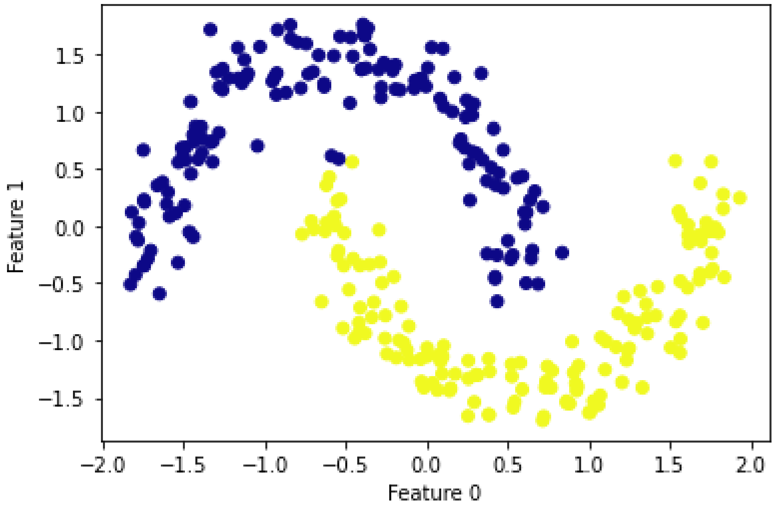
Out[36]:
```
array([1, 0, 2, 0, 0, 0, 2, 2, 3, 3, 2, 0, 3, 2, 2, 2, 0, 3, 1, 0, 0, 3,
       1, 1, 1, 3, 1, 3, 2, 3, 2, 2, 3, 1, 0, 0, 2, 3, 1, 1, 3, 3, 1, 3,
       0, 1, 1, 2, 3, 1, 0, 2, 3, 2, 1, 3, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0,
       0, 2, 0, 0, 0, 3, 2, 3, 1, 1, 2, 0, 1, 0, 3, 2, 1, 2, 3, 3, 1, 0,
       2, 2, 1, 3, 0, 1, 2, 3, 0, 0, 2, 1, 3, 1, 3, 3, 2, 1, 0, 3, 1, 0,
       3, 1, 1, 1, 3, 1, 2, 1, 3, 0, 2, 2, 1, 3, 2, 0, 3, 3, 1, 1, 3, 0,
       0, 3, 1, 3, 1, 3, 3, 3, 1, 0, 3, 1, 0, 3, 3, 0, 1, 0, 1, 3, 0, 1,
       1, 0, 1, 1, 1, 3, 1, 1, 3, 0, 1, 1, 0, 0, 2, 2, 3, 3, 0, 0, 1, 2,
       2, 3, 0, 1, 1, 3, 1, 0, 2, 3, 1, 1, 3, 1, 3, 1, 2, 2, 2, 3, 2, 1,
       2, 0, 1, 1, 3, 1, 0, 3, 3, 0, 2, 3, 0, 1, 2, 2, 0, 2, 1, 2, 2, 2,
       0, 3, 1, 2, 2, 2, 2, 3, 3, 3, 0, 1, 3, 0, 0, 3, 3, 1, 1, 1, 3, 1,
       0, 1, 2, 0, 3, 0, 3, 2, 0, 2, 2, 2, 0, 2, 0, 2, 1, 0, 2, 2, 0, 1,
       3, 1, 0, 0, 2, 3, 0, 3, 0, 0, 3, 2, 1, 1, 3, 1, 2, 1, 1, 3, 1, 2,
       2, 1, 2, 2, 3, 2, 1, 3, 3, 0, 3, 0, 1, 0])
```

In [37]:
```
#visualizing clusters and centroids
sns.scatterplot(x_scaled[:,0], x_scaled[:,1], hue = ["cluster-{}".format(x) for x in y_kmeans])
plt.title("Clusters and Centroids")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.legend()
plt.show()
```

```
C:\Users\meorh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

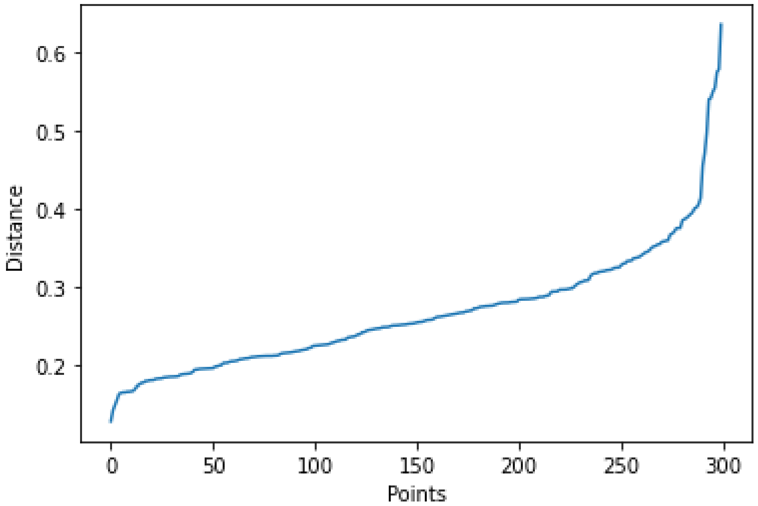


In [38]:
```
#2) Perform DBSCAN clustering (use knee locator to help find optimal parameter) on the given dataset, display cluste
#visualizing using DBSCAN
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs
from sklearn.neighbors import NearestNeighbors
from kneed import KneeLocator

db = DBSCAN(eps=0.5, min_samples=10).fit(x_scaled)
labels = db.labels_
plt.scatter(x_scaled[:, 0], x_scaled[:, 1], c=labels, cmap="plasma")
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.show()
```



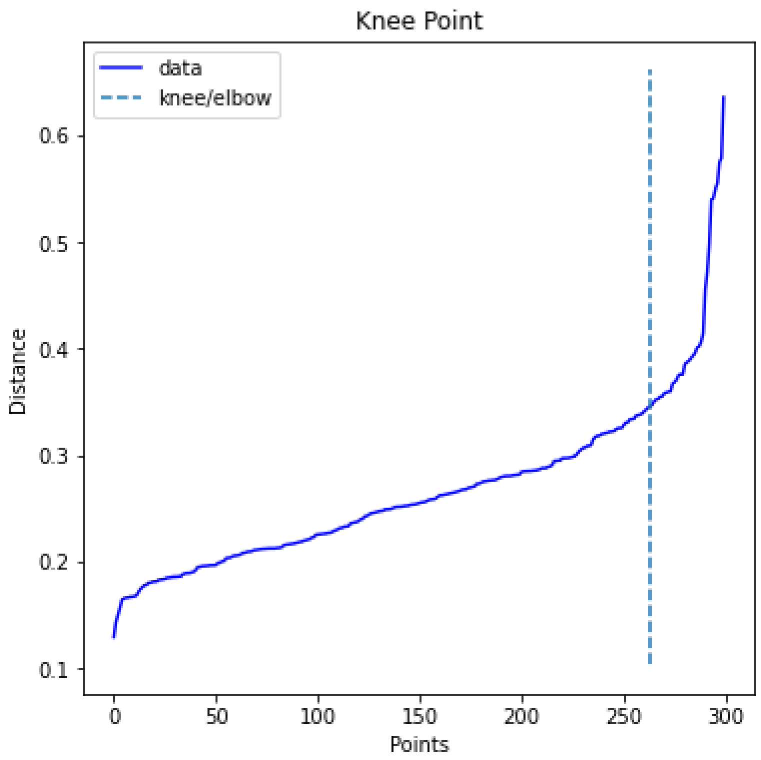
In [39]:
```
#defining neighbour
nearest_neighbors = NearestNeighbors(n_neighbors=11)
neighbors = nearest_neighbors.fit(x_scaled)
distances, indices = neighbors.kneighbors(x_scaled)
```

In [40]:
```
#plotting knee value
distances = np.sort(distances[:,10],axis=0)
i = np.arange(len(distances))
plt.plot(distances)
plt.xlabel('Points')
plt.ylabel('Distance')
plt.show()
```



In [41]:
```
#interpreting kneelocator
knee = KneeLocator(i, distances, S=1, curve='convex',
direction='increasing',interp_method='polynomial')
```

In [42]:
```
#finding real knee value
knee.plot_knee()
plt.xlabel('Points')
plt.ylabel('Distance')
plt.show()
print(distances[knee.knee]) # "just right far"
```

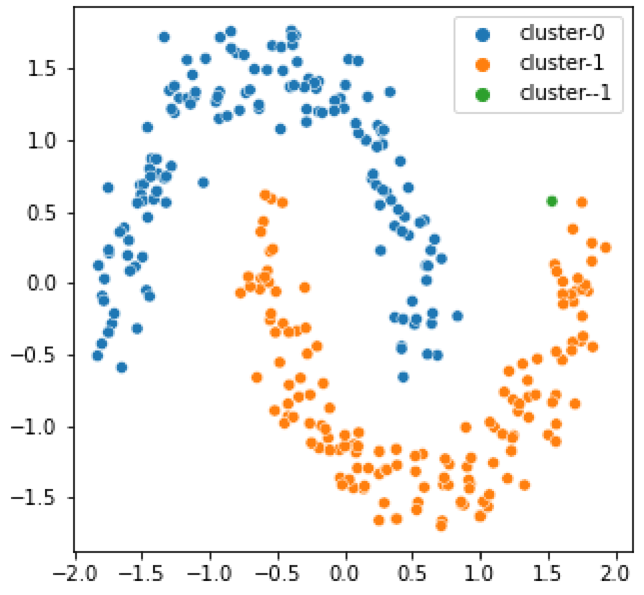



```
0.34567786876134754
```

In [43]:
```
#fine tuning dbscan using knee value
db = DBSCAN(eps=distances[knee.knee], min_samples=10)
db.fit(x_scaled)
labels = db.labels_
fig = plt.figure(figsize=(5,5))
sns.scatterplot(x_scaled[:,0], x_scaled[:,1], hue=["cluster-{}".format(x) for x in labels])
```

```
C:\Users\meorh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[43]: <AxesSubplot:>



3) Conduct comparison studies on the two techniques (K-Means and DBSCAN), with graphical visualization comparisons, discuss your results and decide on whether:
a. K-Means is the better clustering technique for this dataset or,
b. DBSCAN is the better clustering technique for this dataset, or
c. There's no clear distinction between the two techniques for this dataset

```
...
3) Conduct comparison studies on the two techniques (K-Means and DBSCAN)
Discussion on DBSCAN vs KMeans

Comparing between two techniques based on both graphs, clearly DBSCAN showed
better clustering area. This is because the data is in arbitrary shape where DBSCAN relies on density of data
points.
This is a weakness of KMeans technique since KMeans only measures the
radius of its centroid. KMeans also is bad when data
...
```

In [ ]: