chapter 03 Describing data with statistics

In [1]:
```
shortlist = [1,2,3] #findiing the mean
sum(shortlist)
```

Out[1]: 6

In [2]:
```
len(shortlist) #len func give the length of the list
```

Out[2]: 3

calculating teh , mean

In [5]:
```
def calculate_mean(numbers):
  s = sum(numbers)
  N = len(numbers)
  #calculate the mean
  mean = s/N
  return mean


if __name__ =='__main__':
  donations = [100, 60, 70 , 900, 100, 200, 500, 503, 600, 1000, 2000]
  mean = calculate_mean(donations)
  N = len(donations)
  print('mean donation over the last {0} days is {1}'.format(N, mean))
```

```
mean donation over the last 11 days is 548.4545454545455
```

In [6]:
```
# dataset in python find the mean
"""data.mean()"""
```

Out[6]: 'data.mean()'

finding the median

In [7]:
```
samplelist = [4,1,3] #sort()
samplelist.sort()
samplelist
```

Out[7]: [1, 3, 4]

calculating median:

In [9]:
```
def calculate_median(numbers):
  N = len(numbers)
  numbers.sort()

                        #find the median
  if N % 2 == 0:
                         #if N is even
    m1 = N/2
    m2 = (N/2) +1
                         #convert to intgers ,match position
    m1 = int(m1) -1
    m2 = int(m2) -1
    median = (numbers[m1] + numbers[m2])/2
  else:
    m = (N+1)/2
    #convert to integer , match position
    m = int(m) -1
    median = numbers[m]

  return median

if __name__  =='__main__':
  donations = [100,60,900,100,200,500, 500, 503, 600, 1000, 1200]
  median = calculate_median(donations)
  N = len(donations)
  print('median donation over the last {0} days is {1}'.format(N, median))
```

```
median donation over the last 11 days is 500
```

Finding the mode and creating the frequency table

In [10]:
```
simplelist =[4,2,1,3,4]
from collections import Counter
c= Counter(simplelist)
c.most_common()
```

Out[10]:
```
[(4, 2), (2, 1), (1, 1), (3, 1)]
```

In [11]:
```
c.most_common(1)
```

Out[11]:
```
[(4, 2)]
```

In [13]:
```
c.most_common(2)
```

Out[13]:
```
[(4, 2), (2, 1)]
```

In [14]:
```
mode = c.most_common(1)
mode
```

Out[14]:
```
[(4, 2)]
```

In [15]:
```
mode[0]
```

Out[15]:
```
(4, 2)
```

In [16]:
```
mode[0][0]
```

Out[16]:
```
4
```

calculating the mode

In [17]:
```
from collections import Counter

def calculate_mode(numbers):
  c = Counter(numbers)
  mode = c.most_common(1)
  return mode[0][0]


if __name__ =='__main__':
  scores = [7, 8 , 9, 2, 10, 9, 9, 9, 9, 4, 5, 6, 1, 5, 6, 7, 8, 6, 1, 10]
  mode = calculate_mode(scores)

  print('the mode of the list of numbers is :{0}'.format(mode))
```

```
the mode of the list of numbers is :9
```

Calculating the mode wen the list of numbers may have the multiple modes

In [19]:
```
from  collections import Counter

def calculate_mode(numbers):

  c =Counter(numbers)
  numbers_freq =c.most_common()
  max_count = numbers_freq[0][1]
  modes = []

  for num in numbers_freq:
    if num[1] == max_count:
      modes.append(num[0])

  return modes

if __name__ =='__main__':
  scores = [5,5,5,4,4,4,9,1,3]
  modes = calculate_mode(scores)
  print('the mode(s) of the list of the numbers are :')
  for mode in modes:
    print(mode)
```

```
the mode(s) of the list of the numbers are :
5
4
```

creating a frequency table

In [21]:
```
from collections import Counter
                                    # creating the frequency table for a list
of numbers
def frequency_table(numbers):
  table = Counter(numbers)
  print('Number\tFrequency')
  for number in table.most_common():
    print('{0}\t{1}'.format(number[0],number[1]))


if __name__ =='__main__':
  scores = [7,8,9,2,10,9,9,9,9,4,5,6,1,5,6,7,8,6,1,10]
  frequency_table(scores)
```

```
Number  Frequency
9       5
6       3
7       2
8       2
10      2
5       2
1       2
2       1
4       1
```

Frequency table for a list of numbers Enhanced to display the table sorted by the numbers

In [24]:
```
from collections import Counter

def frequency_table(numbers):
  table = Counter(numbers)
  numbers_freq = table.most_common()
  numbers_freq.sort()

  print('number\tFrequency')
  for number in numbers_freq:
    print('{0}\t{1}'.format(number[0], number[1]))


if __name__ == '__main__':
  scores = [7,8,9,2,10,9,9,9,9,4,5,6,1,5,6,7,8,6,1,10]
  frequency_table(scores)
```

```
number  Frequency
1       2
2       1
4       1
5       2
6       3
7       2
8       2
9       5
10      2
```

Measuring the dispersion

finding the range of a set of numbers

In [27]:
```
def find_range(numbers):

  lowest = min(numbers)
  highest = max(numbers)
                              #find the range
  r = highest - lowest

  return lowest , highest ,r


if __name__ == '__main__':
  donations = [100, 60, 70, 900, 100, 200, 500, 500, 503, 600, 1000, 1200]
  lowest , highest, r = find_range(donations)
  print('Lowest:{0} Highest: {1} Range: {2}'.format(lowest, highest, r))
```

```
Lowest:60 Highest: 1200 Range: 1140
```

A high variance means that values are far from the mean; a low variance means that the values are clustered close to the mean. We calculate the variance using the formula

$$\text{variance} = \frac{\sum\left(x_i - x_{\text{mean}}\right)^2}{n}.$$

In the formula, x_i stands for individual numbers (in this case, daily total donations), x_{mean} stands for the mean of these numbers (the mean daily donation), and n is the number of values in the list (the number of days on which donations were received). For each value in the list, we take the difference between that number and the mean and square it. Then, we add all those squared differences together and, finally, divide the whole sum by n to find the variance.

If we want to calculate the standard deviation as well, all we have to do is take the square root of the variance. Values that are within one standard deviation of the mean can be thought of as fairly typical, whereas values that are three or more standard deviations away from the mean can be considered much more atypical—we call such values *outliers*.

Finding the Variance and Standard Deviation

Find the variance and standard deviation of a list of numbers

In [28]:
```
def calculate_mean(numbers):
  s = sum(numbers)
  N = len(numbers)
  #calculate the mean
  mean = s/N

  return mean

def find_differences(numbers):
  #find the mean
  mean = calculate_mean(numbers)
  #find the differnces from the mean
  diff = []
  for num in numbers:
    diff.append(num-mean)

  return diff


def calculate_varience(numbers):
  #find the list of differences
  diff = find_differences(numbers)
  #find the squared differences
  squared_diff = []
  for d in diff:
    squared_diff.append(d**2)
    #find the varience
    sum_squared_diff = sum(squared_diff)
    varience = sum_squared_diff/len(numbers)
    return varience


if __name__ == '__main__':
  donations = [100, 60,70,900, 100,200,500,500, 503,600,1000,1200]
  varience = calculate_varience(donations)
  print('the varience of the list of numbers is {0}'.format(varience))

  std = varience**0.5
  print('the standrd deviation of the list numbers is of numbers is {0}'.forma
t(std))
```

```
the varience of the list of numbers is 11891.255208333334
the standrd deviation of the list numbers is of numbers is 109.04703209319057
```

calculating the correlation btw two data sets

Calculating the Correlation Coefficient

The correlation coefficient is calculated using the formula

$$\text{correlation} = \frac{n\Sigma xy - \Sigma x \Sigma y}{\sqrt{\left(n\Sigma x^2 - (\Sigma x)^2\right)\left(n\Sigma y^2 - (\Sigma y)^2\right)}}.$$

In the above formula, n is the total number of values present in each set of numbers (the sets have to be of equal length). The two sets of numbers are denoted by x and y (it doesn't matter which one you denote as which). The other terms are described as follows:

Σxy	Sum of the products of the individual elements of the two sets of numbers, x and y
Σx	Sum of the numbers in set x
Σy	Sum of the numbers in set y
$(\Sigma x)^2$	Square of the sum of the numbers in set x
$(\Sigma y)^2$	Square of the sum of the numbers in set y
Σx^2	Sum of the squares of the numbers in set x
Σy^2	Sum of the squares of the numbers in set y

In [29]:
```
simple_list1 = [1,2,3]
simple_list2 =[4,5,6]
for x, y in zip(simple_list1, simple_list2):
  print(x,y)
```

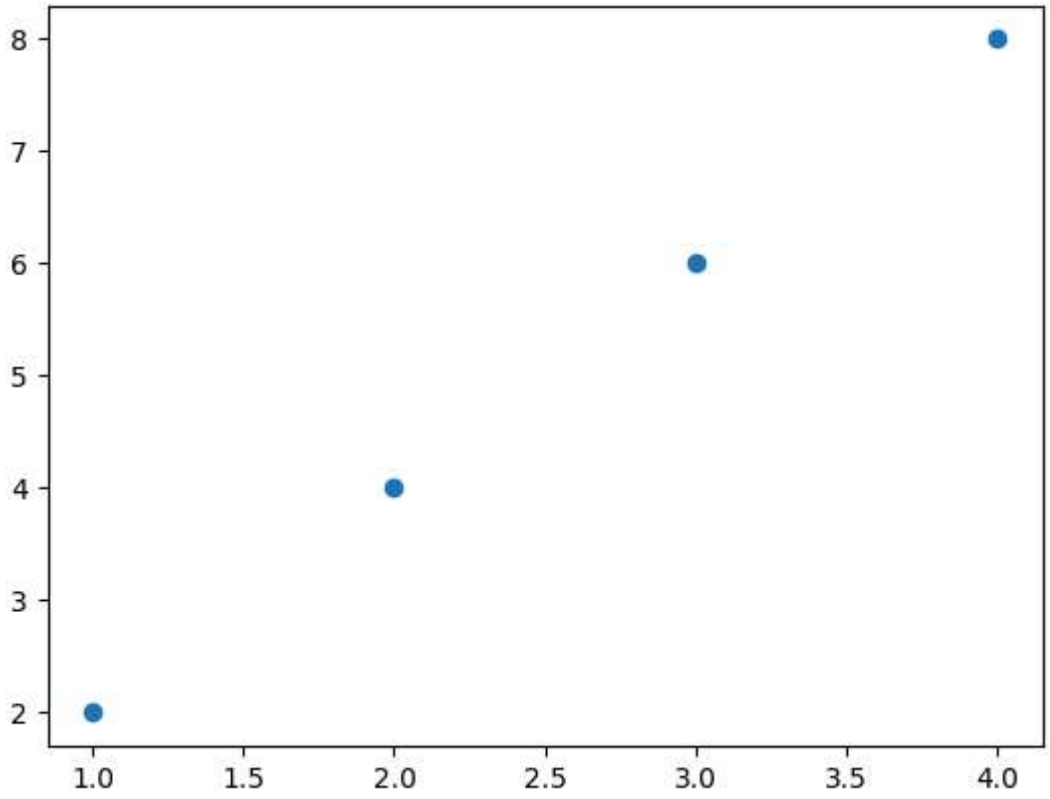
```
1 4
2 5
3 6
```

In [30]:
```
def find_corr_x_y(x,y):
  n = len(x)

  #find the sum of the products
  prod = []
  for xi,yi in zip(x,y):
    prod.append(xi*yi)

  sum_prod_x_y = sum(prod)
  sum_x =sum(x)
  sum_y =sum(y)
  squared_sum_x = sum_x**2
  squared_sum_y = sum_y**2

  x_square =[]
  for xi in x:
    x_square.append(xi**2)
    #find the sum
    x_square_sum = sum(x_square)

    y_square =[]
    for yi in y:
      y_square.append(yi**2)
      #find the sum
      y_square_sum = sum(y_square)

    #use formula to calulate correlation
    numerator = n*sum_prod_x_y - sum_x*sum_y
    denominator_term1 = n*x_square_sum - squared_sum_x
    denominator_term2 = n*y_square_sum - squared_sum_y
    denominator = (denominator_term1*denominator_term2)**0.5
    coorelation = numarator/denominator

    return coorelation
```

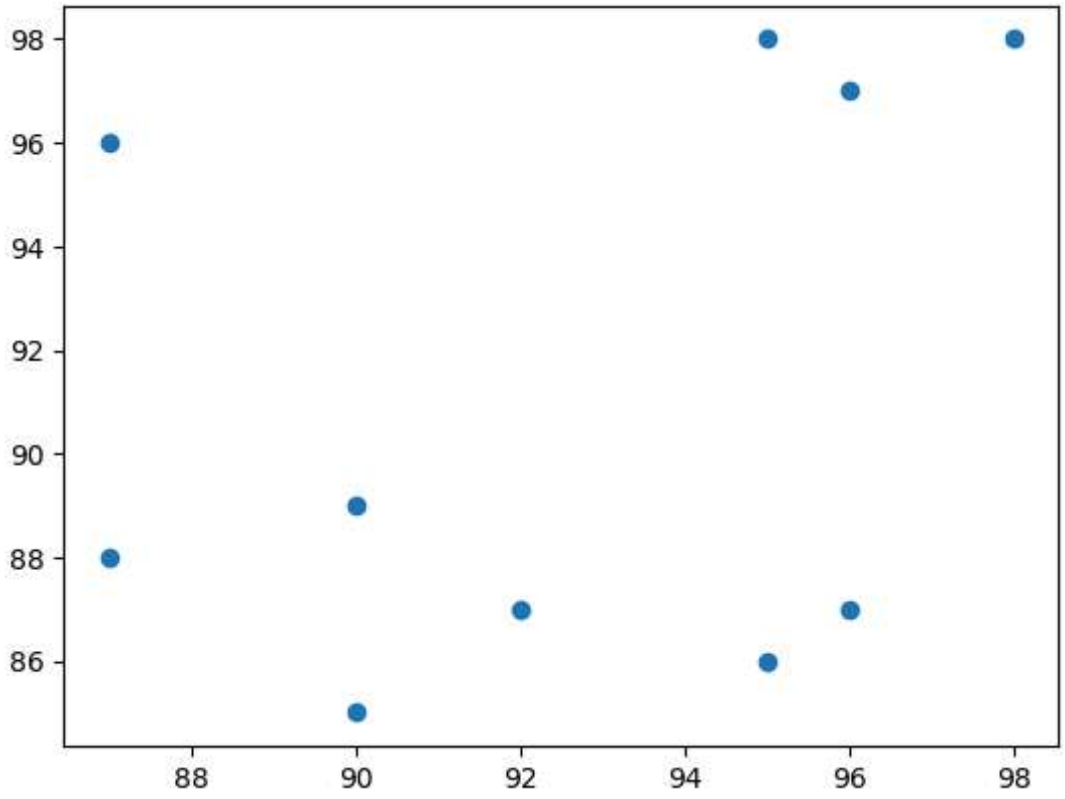
High School Grades and Performance on College Admission Tests

scatter plots

In [32]:
```
x = [1,2,3,4]
y = [2,4,6,8]
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.show()
```



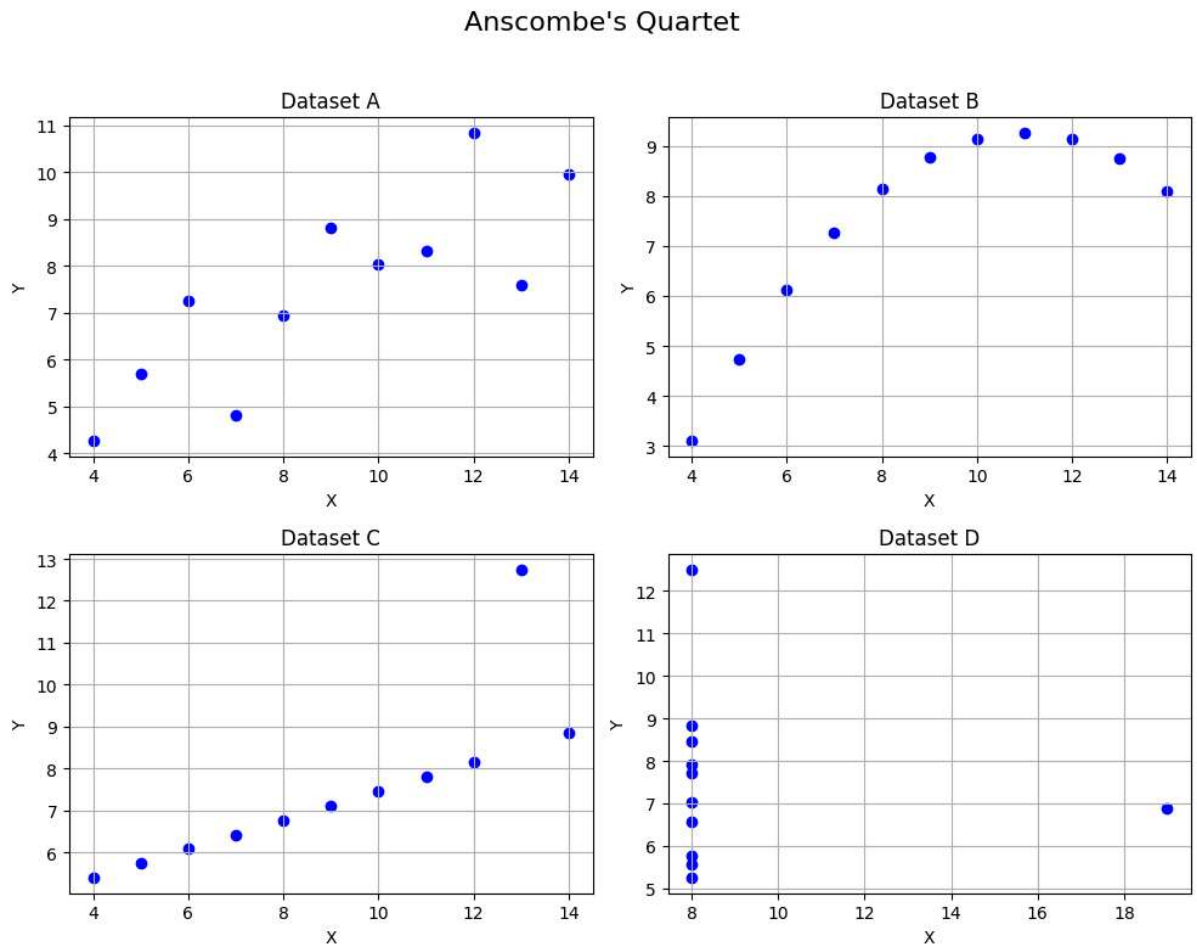
In [34]:
```
r = [90,92,95,96,87,87,90,95,98,96]
m = [85,87,86,97,96,88,89,98,98,87]
plt.scatter(r,m)
plt.show()
```



In [35]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Data for Anscombe's Quartet
data = {
    "A": {
        "X": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
        "Y": [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.6
8],
    },
    "B": {
        "X": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
        "Y": [9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.7
4],
    },
    "C": {
        "X": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
        "Y": [7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.7
3],
    },
    "D": {
        "X": [8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 19.0],
        "Y": [6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.8
9],
    },
}

# Function to compute statistical measures
def compute_statistics(x, y):
    mean_x = np.mean(x)
    std_x = np.std(x, ddof=1)
    mean_y = np.mean(y)
    std_y = np.std(y, ddof=1)
    correlation = np.corrcoef(x, y)[0, 1]
    return mean_x, std_x, mean_y, std_y, correlation

# Display statistics for each dataset
for dataset_name, dataset in data.items():
    x, y = dataset["X"], dataset["Y"]
    mean_x, std_x, mean_y, std_y, correlation = compute_statistics(x, y)
    print(f"Dataset {dataset_name}:")
    print(f"  Mean X: {mean_x:.2f}, Std Dev X: {std_x:.2f}")
    print(f"  Mean Y: {mean_y:.2f}, Std Dev Y: {std_y:.2f}")
    print(f"  Correlation: {correlation:.2f}\n")

# Plot scatter plots for each dataset
fig, axes = plt.subplots(2, 2, figsize=(10, 8))
fig.suptitle("Anscombe's Quartet", fontsize=16)

for (dataset_name, dataset), ax in zip(data.items(), axes.flatten()):
    x, y = dataset["X"], dataset["Y"]
    ax.scatter(x, y, label=dataset_name, color="blue")
    ax.set_title(f"Dataset {dataset_name}")
    ax.set_xlabel("X")
    ax.set_ylabel("Y")
```

```
    ax.grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

```
Dataset A:
  Mean X: 9.00, Std Dev X: 3.32
  Mean Y: 7.50, Std Dev Y: 2.03
  Correlation: 0.82

Dataset B:
  Mean X: 9.00, Std Dev X: 3.32
  Mean Y: 7.50, Std Dev Y: 2.03
  Correlation: 0.82

Dataset C:
  Mean X: 9.00, Std Dev X: 3.32
  Mean Y: 7.50, Std Dev Y: 2.03
  Correlation: 0.82

Dataset D:
  Mean X: 9.00, Std Dev X: 3.32
  Mean Y: 7.50, Std Dev Y: 2.03
  Correlation: -0.10
```



Reading Data from a Text File

1. creating the txt file with data
   A. uploading and loading the google colab
   B. copypath of the uploaded file

In [36]:
```
def sum_data(filename):
  s = 0
  with open(filename) as f:
    for line in f:
      s =s + float(line)
  print('sum of the numbers:{0}'.format(s))

if __name__ == '__main__':
  sum_data('/content/data.txt')
```

```
sum of the numbers:5733.0
```

Calculating the mean of numbers stored in a file

In [37]:
```
def read_data(filename):

  numbers = []
  with open(filename) as f:
    for line in f:
      numbers.append(float(line))


      return numbers

def calulate_mean(numbers):
  s = sum(numbers)
  N = len(numbers)
  mean = s/N

  return mean


if __name__ == '__main__':
  data = read_data('/content/data.txt')
  mean = calculate_mean(data)
  print('mean: {0}'.format(mean))
```

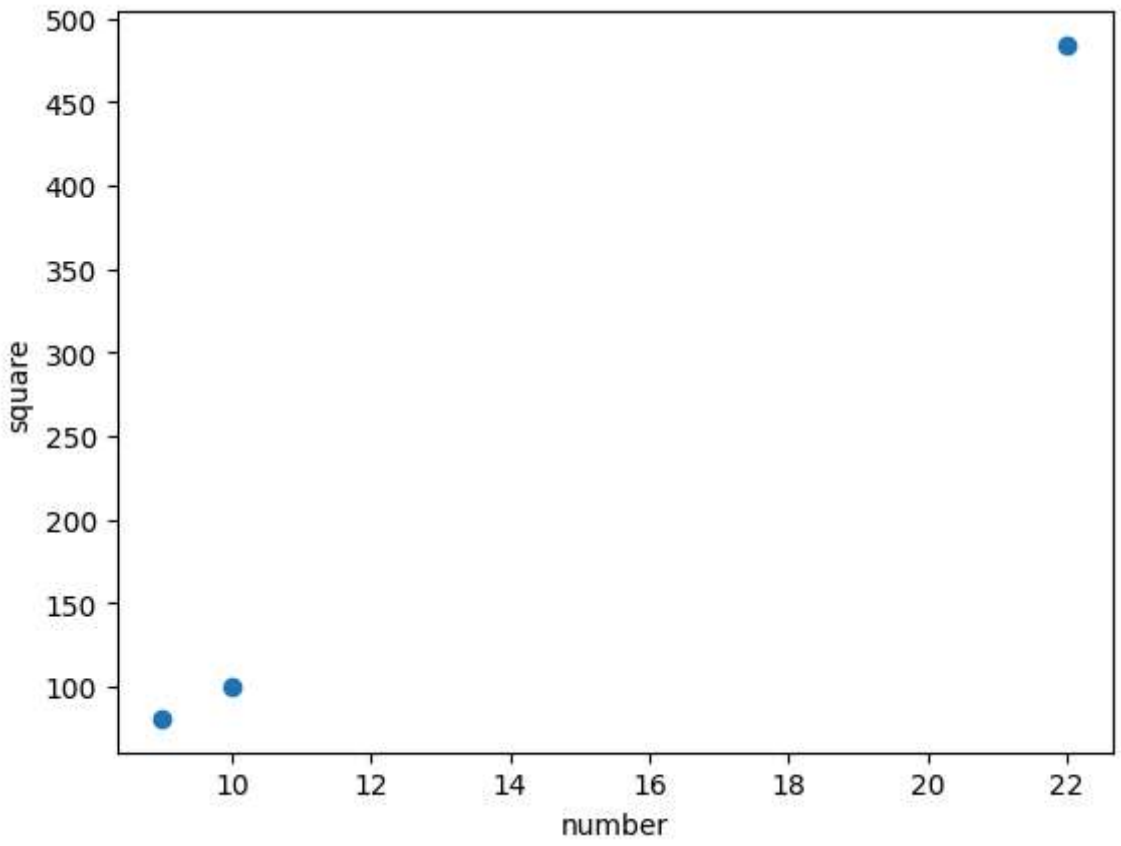
```
mean: 100.0
```

Reading Data from a CSV File

comma seperated values which are formated in the form of rows nd coulmns

- creating the csv file
- upload google colab

In [39]:
```
import csv
import matplotlib.pyplot as plt

def scatter_plot(x,y):
  plt.scatter(x,y)
  plt.xlabel('number')
  plt.ylabel('square')
  plt.show()

def read_csv(filename):
  numbers = []
  squared =[]
  with open(filename) as f:
    reader = csv.reader(f)
    next(reader)
    for row in reader:
      numbers.append(float(row[0]))
      squared.append(float(row[1]))
  return numbers, squared

if __name__ == '__main__':
  numbers, squared = read_csv('/content/number.csv')
  scatter_plot(numbers, squared)
```



In [40]:
```
#reading a csv file in easy way
import pandas as pd
df = pd.read_csv('/content/number.csv') # any other csv files
df.head()
```

Out[40]:

	Number	Squared
0	10	100
1	9	81
2	22	484

chapter ends.

In [ ]:
```

```