Q.1) Program:-

```
i=1
while<11;
    Print(i)
    i=i+1
```

Q.2)

```
n= 4
For i in range(n):
    For j in range(i+1):
        Print("  ", end=" ")
    for j in range(i,n):
        Print("1", end=" ")
    Print()
```

Q.3)

```
num=5
Sum=0
For i in range(num):
    Sum=Sum+i

Print(Sum)
```

Q.4)

```
T= int(input("enter the number")
i=1
while i<=10:
    Print(T,'x',i,'=',i*table)
    i=i+1
```

Q.5)

```
li=[12,13,14,15,16,17]
For i in (list)
    Print(i)
    i=i+1
```

Q.6)

```
num=123456
Print(len(str(num)))
```

Q.8)

```
01=[1,2,3,4,5]
R1=[]
For i in 01:
    R1=[i]+R1
Print(R1)
```

Q.9)

```
For i in range(-10,0):
    Print(i, end=" ")
```

10)

```
n=6
if n<5:
    Print("notdone")
else:
    Print("Done")
```

Q.11)

```
num=11
For i in range(2,num):
    if num%i==0:
        Print("not Prime")
        break
else:
    Print("Prime")
```

)

```
num=10
a=0
b=1
Print(a)
Print(b)
For i inrange(2,num)
    c=a+b
    a=b
    b=c
Print(c)
```

Q.13)

```
n=int(input("enterthenumber")
Fact=1
if n>=1:
    For i inrange (1,n+1)
        fact=fact*i
Print("factorialof a given number," fact)
Q.14) num = 234
      r_num = 0
      while num != 0:
            digit = num%10
            r_num = r_num *10 + digit
            num// = 10
        Print(" Reversed Number" + str(r_num))
```

```
Q.15) list = [31,42,13,34,85,0,99,1,3]
      For i in range(1, len(list),2):
            Print(list)
```

```
Q.16) n = 8
      For i in range(0,n+1):
            i = i+1
            Print (i*i*i)
```

```
Q.17) num = 10
      sum = int (num*(num+1))/2
      Print (sum)
```

```
Q.7) n=5
     For x in range(n):
           Print ("   " * (n-x), "*" * (2*x+1))

     For x in range(n-2,-1,-1):
           Print("   " *(n-x),"*" * (2*x+1))
```

Assignment NO:-2

1) S.d. = 1.5   Data = 2, 3, 1, 3, 2, 4

$$\mu = \frac{2+3+1+3+2+4}{6} = 2.5$$

Formula for z-score $= \frac{x-\mu}{\sigma}$

I) For x=2 $\quad \frac{2-2.5}{1.5} = -0.33$

II) For x=3 $\quad \frac{3-2.5}{1.5} = 0.33$

III) For x=1 $\quad \frac{1-2.5}{1.5} = -1$

IV) For x=3 $\quad \frac{3-2.5}{1.5} = 0.33$

V) For x=2 $\quad \frac{2-2.5}{1.5} = -0.33$

VI) For x=4 $\quad \frac{4-2.5}{1.5} = 1$

Formula For normalization:-

$$x_{nor} = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

2) one-hot encoding:- OHE is a method used to represent categori variable as binary ~~vet~~ vectors. It is used when machine learning model may not be able to work directly with categorical data.

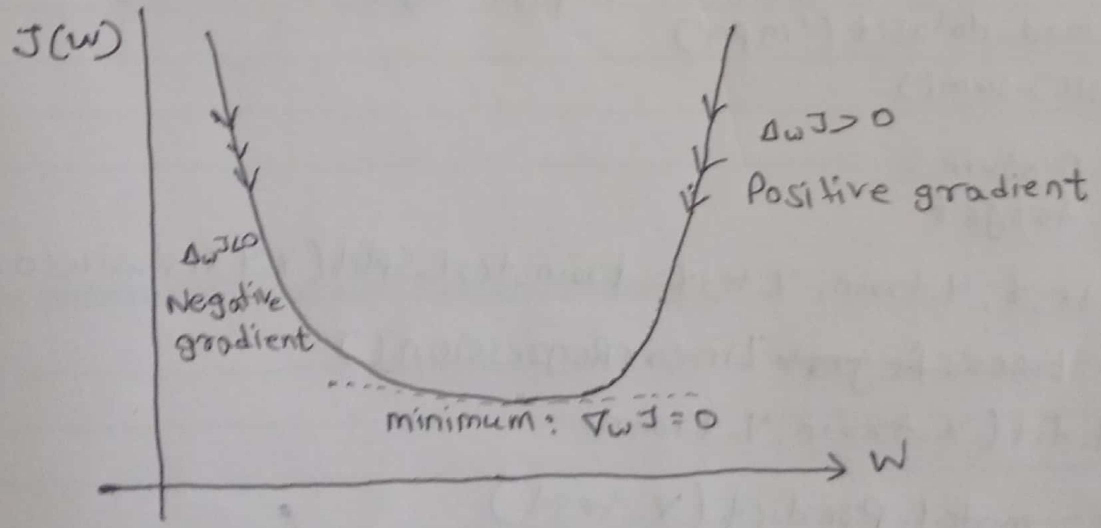
Pandas function to Perrfrom OHE is called 'get-dumies'.

3) (I) log transformer (II) Reciprocal transformer (III) Square (IV) square root
(V) custom

Power transformer
(I) Box cox (II) Yeo Johnson.

(4) Assumption of linear regression :-
(I) Linearity assume that the relationship between the independent variable (x) & dependent variable (Y) is linear.
(II) multi variate Normality :- residual values in regression are normally distributed.
(III) Homoscedasticity :- Same variance.
(IV) Independence :- assume that independent variables in model are not correlated or related to one another.

(5) Gradient Descent :- is the most common optimization algorithm in machine learning and deep learning. It is a first-order optimization algorithm. The size of the step we take on each iteration to reach the local minima is determine by the learning rate α. therefore we follow the direction of the slope downhill ~~until~~ until we have reach local minima.

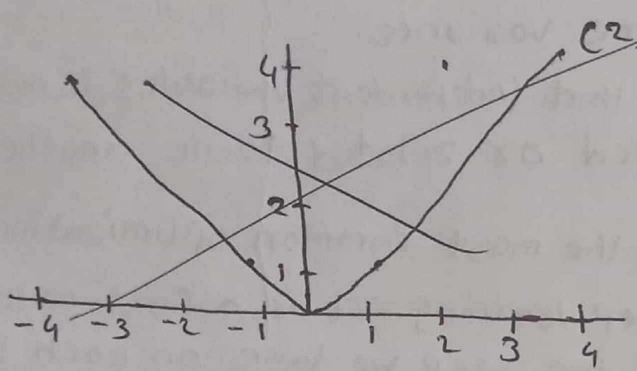


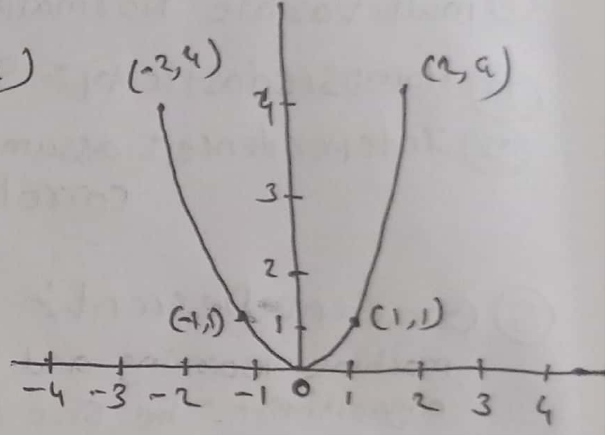
⑥ Pandas Profiling is a python ~~liba~~ library that Provides an overview of a dataframe, ~~incut~~ inculding descriptive ~~Stats~~ statistics, visualization and insights into the distribution of data. It generates an HTML report containing all the information, which makes it easy to explore and understand the structure and characteristics of your data.

```
Code :- import Pandas as pd
        From Pandas_Profiling import ProfileReport.

        df = pd.read_csv ('data.csv')
        Profile = ProfileReport
        Profile.to_file ('output.html')
```

⑦ $Y = x^2$





```
⑧ Import Pandas as ~~pd~~ Pb
Ⓘ Import seaborn as sns
  Import matplotlib.pyplot as plt.
  from sklearn.model_selection import train_test_split.
  From sklearn.model_selection import ~~logisti~~ LinearRegression
Ⓘ df=sns.load_dataset ('mpg')
Ⅲ df.isnull().sum()

Ⅳ x = df.features
  y = df.target

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state
                                                                        =42)
Ⓥ model = ~~LinearRegrat~~ LinearRegression()
  model.fit(x_train, y_train)

Ⓥ y_pred = model.predict(x_test)
  y_pred.
```