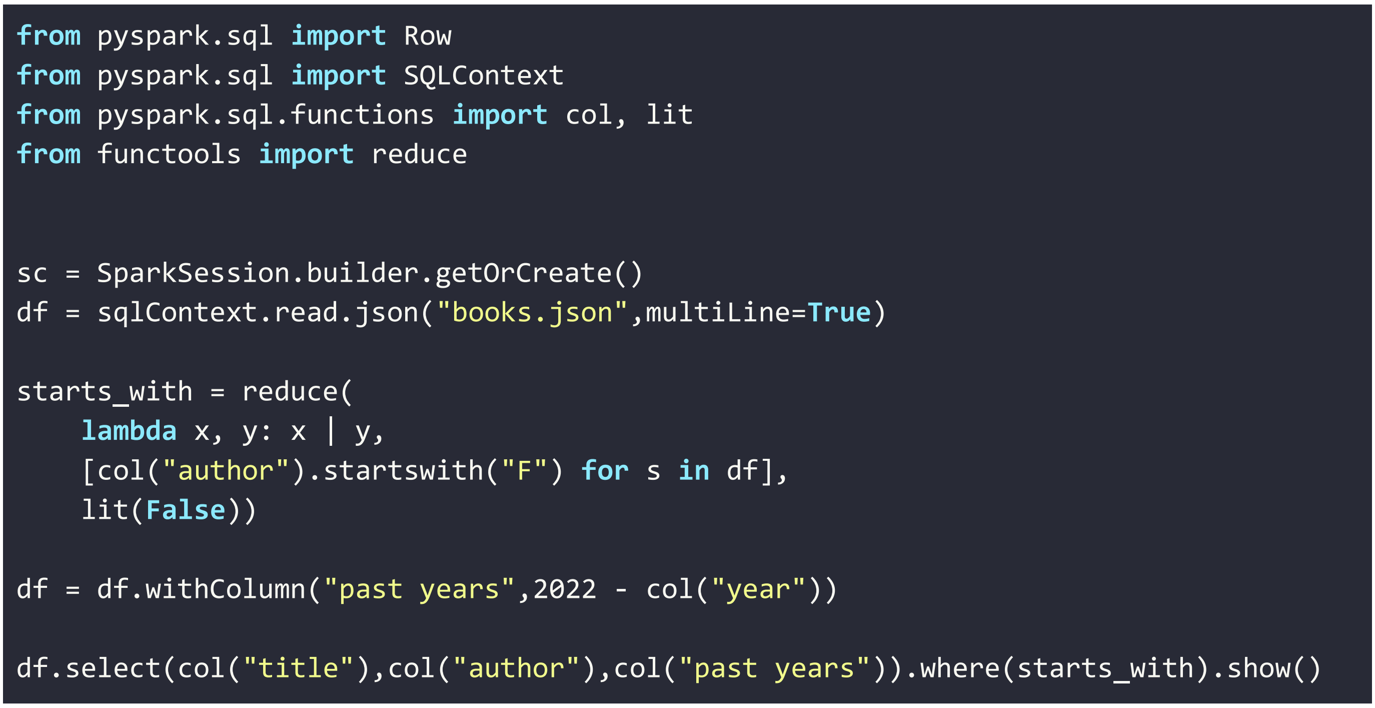
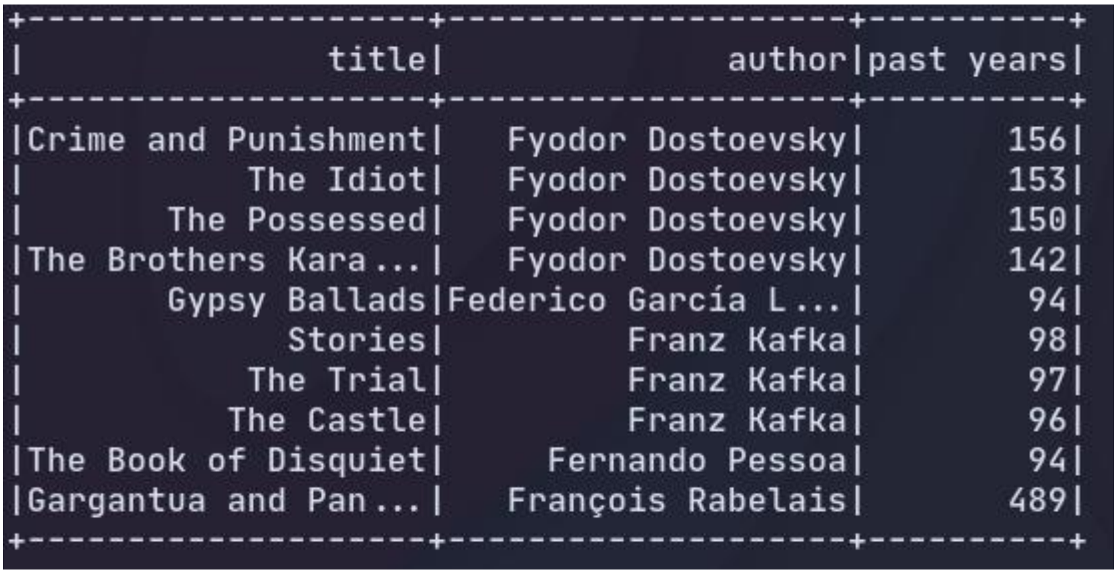
SQL4:

1 - SARK:

a)

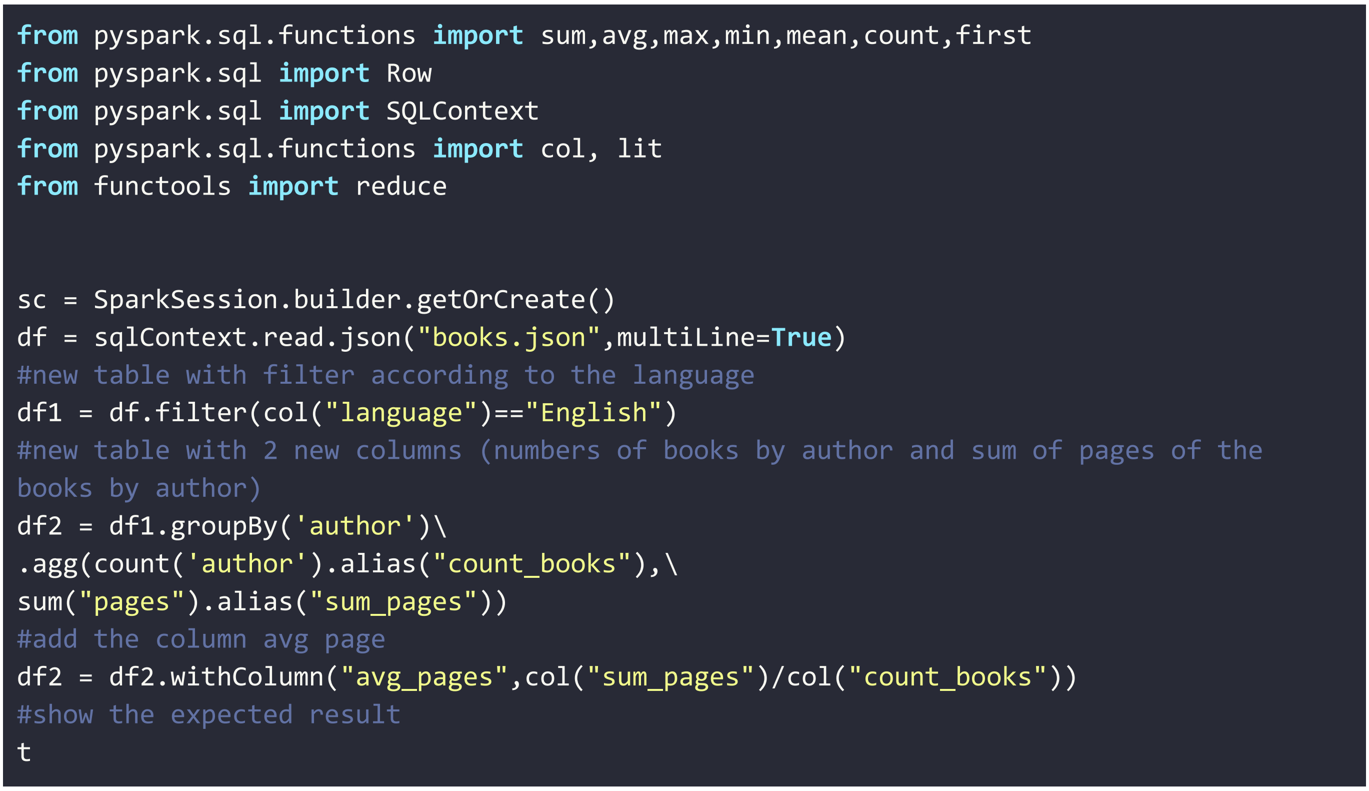


```
from pyspark.sql import Row
from pyspark.sql import SQLContext
from pyspark.sql.functions import col, lit
from functools import reduce



sc = SparkSession.builder.getOrCreate()
df = sqlContext.read.json("books.json",multiLine=True)

starts_with = reduce(
    lambda x, y: x | y,
    [col("author").startswith("F") for s in df],
    lit(False))

df = df.withColumn("past years",2022 - col("year"))

df.select(col("title"),col("author"),col("past years")).where(starts_with).show()
```

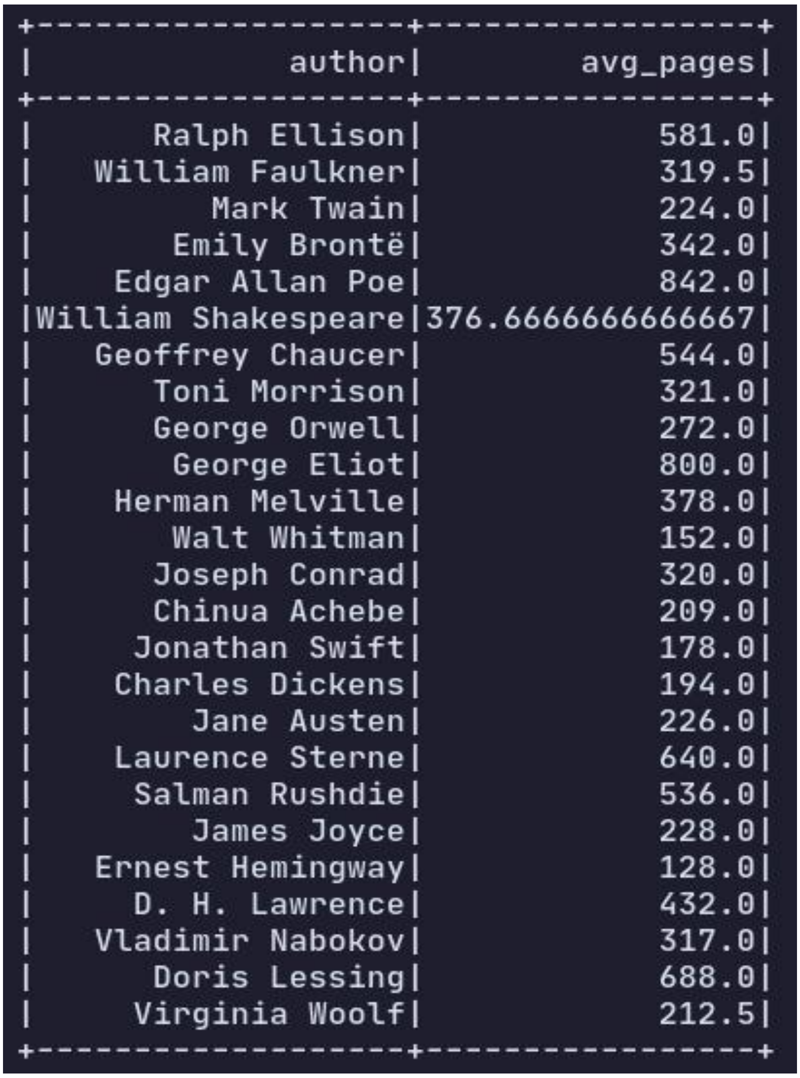


```
+--------------------+--------------------+----------+
|               title|              author|past years|
+--------------------+--------------------+----------+
|Crime and Punishment|   Fyodor Dostoevsky|       156|
|           The Idiot|   Fyodor Dostoevsky|       153|
|       The Possessed|   Fyodor Dostoevsky|       150|
|The Brothers Kara...|   Fyodor Dostoevsky|       142|
|       Gypsy Ballads|Federico García L...|        94|
|             Stories|         Franz Kafka|        98|
|           The Trial|         Franz Kafka|        97|
|          The Castle|         Franz Kafka|        96|
|The Book of Disquiet|     Fernando Pessoa|        94|
|Gargantua and Pan...|   François Rabelais|       489|
+--------------------+--------------------+----------+
```

b)



```
from pyspark.sql.functions import sum,avg,max,min,mean,count,first
from pyspark.sql import Row
from pyspark.sql import SQLContext
from pyspark.sql.functions import col, lit
from functools import reduce


sc = SparkSession.builder.getOrCreate()
df = sqlContext.read.json("books.json",multiLine=True)
#new table with filter according to the language
df1 = df.filter(col("language")=="English")
#new table with 2 new columns (numbers of books by author and sum of pages of the
books by author)
df2 = df1.groupBy('author')\
.agg(count('author').alias("count_books"),\
sum("pages").alias("sum_pages"))
#add the column avg page
df2 = df2.withColumn("avg_pages",col("sum_pages")/col("count_books"))
#show the expected result
t
```

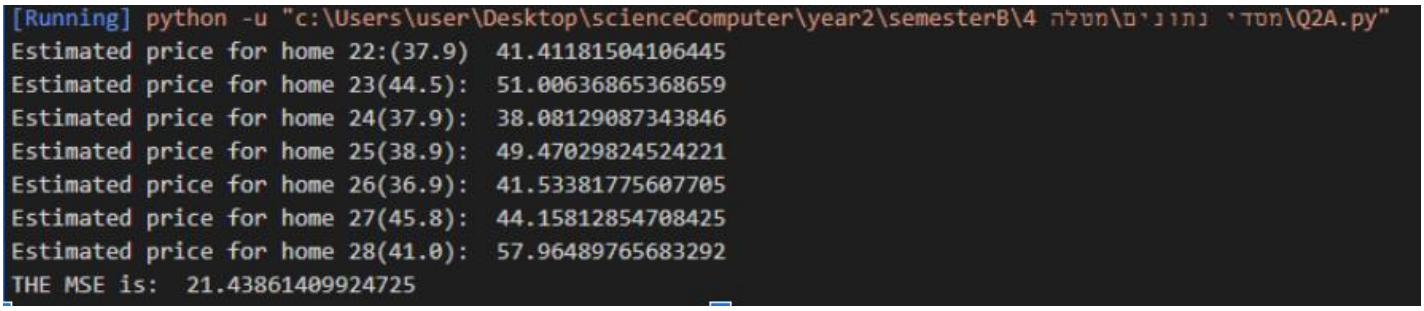


```
+-------------------+-----------------+
|             author|        avg_pages|
+-------------------+-----------------+
|      Ralph Ellison|            581.0|
|   William Faulkner|            319.5|
|         Mark Twain|            224.0|
|       Emily Brontë|            342.0|
|    Edgar Allan Poe|            842.0|
|William Shakespeare|376.6666666666667|
|   Geoffrey Chaucer|            544.0|
|      Toni Morrison|            321.0|
|      George Orwell|            272.0|
|       George Eliot|            800.0|
|    Herman Melville|            378.0|
|       Walt Whitman|            152.0|
|      Joseph Conrad|            320.0|
|      Chinua Achebe|            209.0|
|     Jonathan Swift|            178.0|
|    Charles Dickens|            194.0|
|        Jane Austen|            226.0|
|    Laurence Sterne|            640.0|
|     Salman Rushdie|            536.0|
|        James Joyce|            228.0|
|   Ernest Hemingway|            128.0|
|     D. H. Lawrence|            432.0|
|   Vladimir Nabokov|            317.0|
|      Doris Lessing|            688.0|
|     Virginia Woolf|            212.5|
+-------------------+-----------------+
```

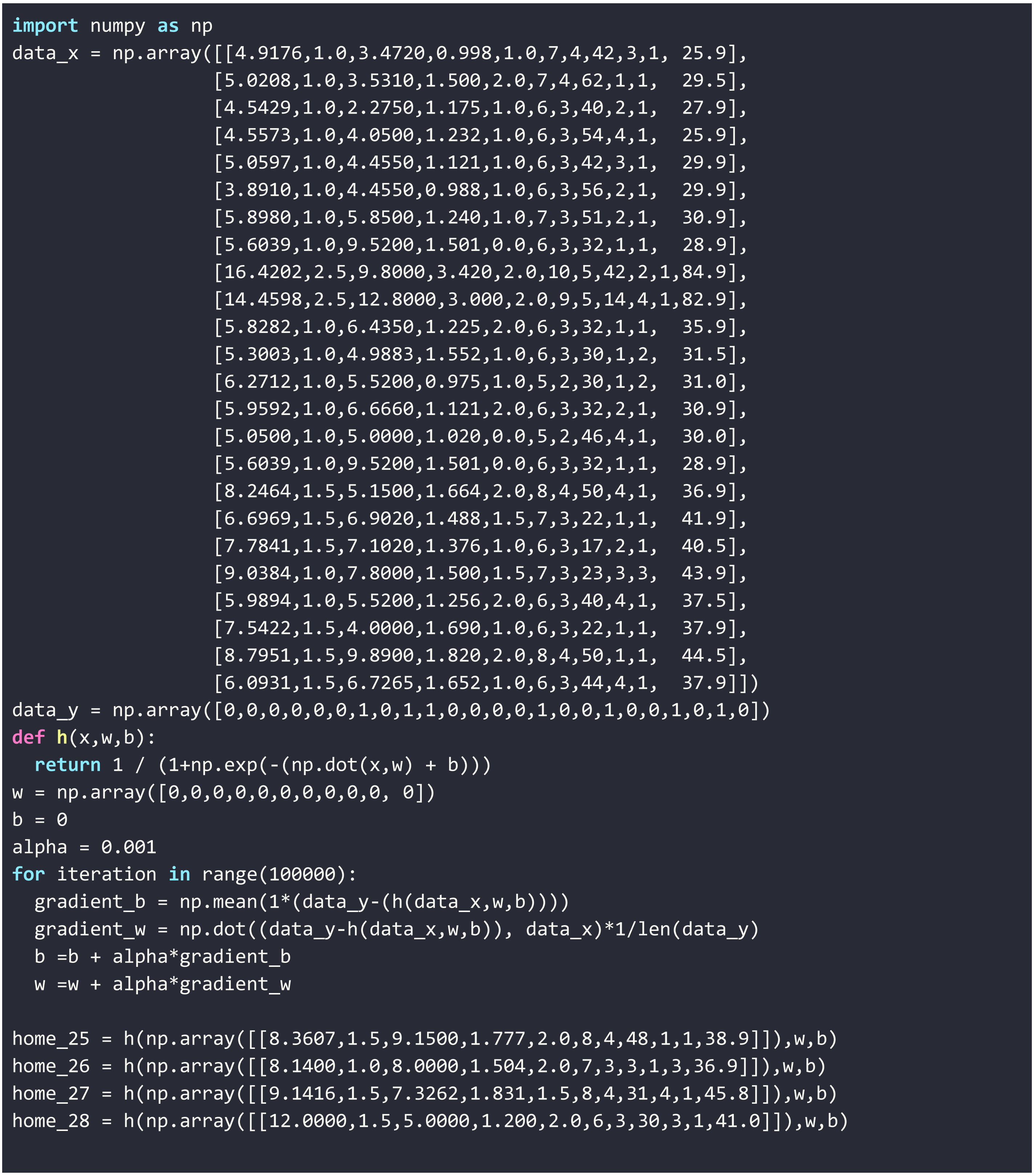
Q2.
1,2)

```
import numpy as np

data_x = np.array([[4.9176,1.0,3.4720,0.998,1.0,7,4,42,3,1,0],
[5.0208,1.0,3.5310,1.500,2.0,7,4,62,1,1,0],
[4.5429,1.0,2.2750,1.175,1.0,6,3,40,2,1,0],
[4.5573,1.0,4.0500,1.232,1.0,6,3,54,4,1,0],
[5.0597,1.0,4.4550,1.121,1.0,6,3,42,3,1,0],
[3.8910,1.0,4.4550,0.988,1.0,6,3,56,2,1,0],
[5.8980,1.0,5.8500,1.240,1.0,7,3,51,2,1,1],
[5.6039,1.0,9.5200,1.501,0.0,6,3,32,1,1,0],
[16.4202,2.5,9.8000,3.420,2.0,10,5,42,2,1,1],
[14.4598,2.5,12.8000,3.000,2.0,9,5,14,4,1,1],
[5.8282,1.0,6.4350,1.225,2.0,6,3,32,1,1,0],
[5.3003,1.0,4.9883,1.552,1.0,6,3,30,1,2,0],
[6.2712,1.0,5.5200,0.975,1.0,5,2,30,1,2,0],
[5.9592,1.0,6.6660,1.121,2.0,6,3,32,2,1,0],
[5.0500,1.0,5.0000,1.020,0.0,5,2,46,4,1,1],
[5.6039,1.0,9.5200,1.501,0.0,6,3,32,1,1,0],
[8.2464,1.5,5.1500,1.664,2.0,8,4,50,4,1,0],
[6.6969,1.5,6.9020,1.488,1.5,7,3,22,1,1,1],
[7.7841,1.5,7.1020,1.376,1.0,6,3,17,2,1,0],
[9.0384,1.0,7.8000,1.500,1.5,7,3,23,3,3,0],
[5.9894,1.0,5.5200,1.256,2.0,6,3,40,4,1,1]])
data_y =
np.array([25.9,29.5,27.9,25.9,29.9,29.9,30.9,28.9,84.9,82.9,35.9,31.5,31.0,30.9,30.0,28.9,
36.9,41.9,40.5,43.9,37.5])
w1 = 0
```

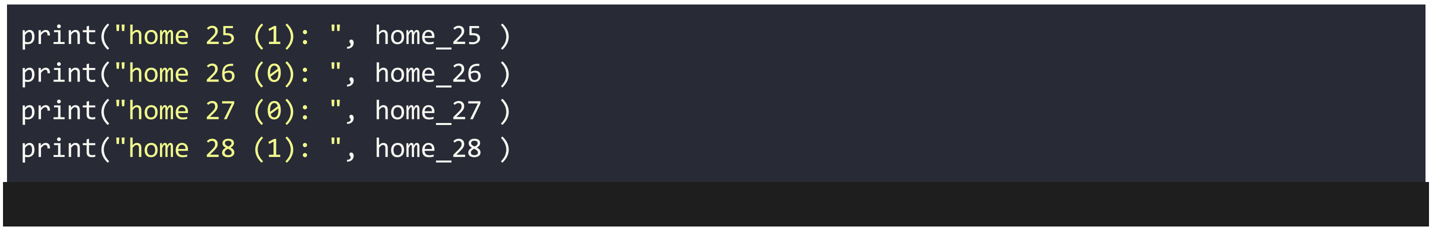
```
w2 = 0
w3 = 0
w4 = 0
w5 = 0
w6 = 0
w7 = 0
w8 = 0
w9 = 0
w10 = 0
w11 = 0
b  = 0
alpha = 0.001
for iteration in range(100000):
    deriv_b =
np.mean(1*((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*
data_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+
b)-data_y))
    deriv_w1 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,0]) * 1.0/len(data_y)
    deriv_w2 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,1]) * 1.0/len(data_y)
    deriv_w3 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,2]) * 1.0/len(data_y)
    deriv_w4 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,3]) * 1.0/len(data_y)
    deriv_w5 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,4]) * 1.0/len(data_y)
    deriv_w6 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,5]) * 1.0/len(data_y)
    deriv_w7 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,6]) * 1.0/len(data_y)
    deriv_w8 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,7]) * 1.0/len(data_y)
    deriv_w9 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,8]) * 1.0/len(data_y)
    deriv_w10 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y), data_x[:,9])* 1.0/len(data_y)
    deriv_w11 =
np.dot(((w1*data_x[:,0]+w2*data_x[:,1]+w3*data_x[:,2]+w4*data_x[:,3]+w5*data_x[:,4]+w6*dat
a_x[:,5]+w7*data_x[:,6]+w8*data_x[:,7]+w9*data_x[:,8]+w10*data_x[:,9]+w11*data_x[:,10]+b)-
data_y),data_x[:,10])* 1.0/len(data_y)
    b -= alpha * deriv_b
    w1 -= alpha * deriv_w1
    w2 -= alpha * deriv_w2
    w3 -= alpha * deriv_w3
    w4 -= alpha * deriv_w4
    w5 -= alpha * deriv_w5
    w6 -= alpha * deriv_w6
    w7 -= alpha * deriv_w7
    w8 -= alpha * deriv_w8
    w9 -= alpha * deriv_w9
    w10 -= alpha * deriv_w10
    w11 -= alpha * deriv_w11

home_22= np.dot(np.array([7.5422 ,1.5,4.0000,1.690,1.0,6,3,22,1,1,0]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b
home_23= np.dot(np.array([8.7951 ,1.5,9.8900,1.820,2.0,8,4,50,1,1,1]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b
home_24= np.dot(np.array([6.0931 ,1.5,6.7265,1.652,1.0,6,3,44,4,1,0]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b
home_25= np.dot(np.array([8.3607 ,1.5,9.1500,1.777,2.0,8,4,48,1,1,1]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b
home_26= np.dot(np.array([8.1400 ,1.0,8.0000,1.504,2.0,7,3,3 ,1,3,0]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b
home_27= np.dot(np.array([9.1416 ,1.5,7.3262,1.831,1.5,8,4,31,4,1,0]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b
home_28= np.dot(np.array([12.0000,1.5,5.0000,1.200,2.0,6,3,30,3,1,1]),np.array([w1, w2,
w3, w4, w5, w6, w7, w8, w9, w10, w11])) + b



actual = [37.9,44.5,37.9,38.9,36.9,45.8,41.0]
forcast = [home_22,home_23,home_24,home_25,home_26,home_27,home_28]
sum=0
for i in (0,6):
    sum=sum+(actual[i]-forcast[i])**2

mse=1/14*(sum)






print("Estimated price for home 22:(37.9) ",home_22 )
print("Estimated price for home 23(44.5): ",home_23 )
print("Estimated price for home 24(37.9): ",home_24 )
print("Estimated price for home 25(38.9): ",home_25 )
print("Estimated price for home 26(36.9): ",home_26 )
print("Estimated price for home 27(45.8): ",home_27 )
print("Estimated price for home 28(41.0): ",home_28 )



print("THE MSE is: ",mse)
```



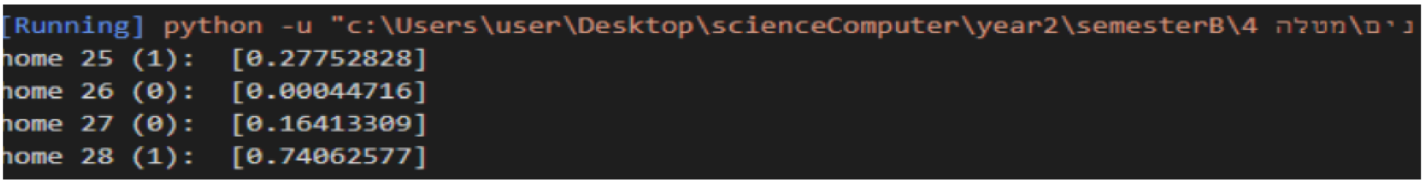
```
[Running] python -u "c:\Users\user\Desktop\scienceComputer\year2\semesterB\4 מטלה\נתונים מסדי\Q2A.py"
Estimated price for home 22:(37.9)  41.41181504106445
Estimated price for home 23(44.5):  51.00636865368659
Estimated price for home 24(37.9):  38.08129087343846
Estimated price for home 25(38.9):  49.47029824524221
Estimated price for home 26(36.9):  41.53381775607705
Estimated price for home 27(45.8):  44.15812854708425
Estimated price for home 28(41.0):  57.96489765683292
THE MSE is:  21.43861409924725
```



```
import numpy as np
data_x = np.array([[4.9176,1.0,3.4720,0.998,1.0,7,4,42,3,1, 25.9],
                  [5.0208,1.0,3.5310,1.500,2.0,7,4,62,1,1,  29.5],
                  [4.5429,1.0,2.2750,1.175,1.0,6,3,40,2,1,  27.9],
                  [4.5573,1.0,4.0500,1.232,1.0,6,3,54,4,1,  25.9],
                  [5.0597,1.0,4.4550,1.121,1.0,6,3,42,3,1,  29.9],
                  [3.8910,1.0,4.4550,0.988,1.0,6,3,56,2,1,  29.9],
                  [5.8980,1.0,5.8500,1.240,1.0,7,3,51,2,1,  30.9],
                  [5.6039,1.0,9.5200,1.501,0.0,6,3,32,1,1,  28.9],
                  [16.4202,2.5,9.8000,3.420,2.0,10,5,42,2,1,84.9],
                  [14.4598,2.5,12.8000,3.000,2.0,9,5,14,4,1,82.9],
                  [5.8282,1.0,6.4350,1.225,2.0,6,3,32,1,1,  35.9],
                  [5.3003,1.0,4.9883,1.552,1.0,6,3,30,1,2,  31.5],
                  [6.2712,1.0,5.5200,0.975,1.0,5,2,30,1,2,  31.0],
                  [5.9592,1.0,6.6660,1.121,2.0,6,3,32,2,1,  30.9],
                  [5.0500,1.0,5.0000,1.020,0.0,5,2,46,4,1,  30.0],
                  [5.6039,1.0,9.5200,1.501,0.0,6,3,32,1,1,  28.9],
                  [8.2464,1.5,5.1500,1.664,2.0,8,4,50,4,1,  36.9],
                  [6.6969,1.5,6.9020,1.488,1.5,7,3,22,1,1,  41.9],
                  [7.7841,1.5,7.1020,1.376,1.0,6,3,17,2,1,  40.5],
                  [9.0384,1.0,7.8000,1.500,1.5,7,3,23,3,3,  43.9],
                  [5.9894,1.0,5.5200,1.256,2.0,6,3,40,4,1,  37.5],
                  [7.5422,1.5,4.0000,1.690,1.0,6,3,22,1,1,  37.9],
                  [8.7951,1.5,9.8900,1.820,2.0,8,4,50,1,1,  44.5],
                  [6.0931,1.5,6.7265,1.652,1.0,6,3,44,4,1,  37.9]])
data_y = np.array([0,0,0,0,0,0,1,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,1,0])
def h(x,w,b):
  return 1 / (1+np.exp(-(np.dot(x,w) + b)))
w = np.array([0,0,0,0,0,0,0,0,0,0, 0])
b = 0
alpha = 0.001
for iteration in range(100000):
  gradient_b = np.mean(1*(data_y-(h(data_x,w,b))))
  gradient_w = np.dot((data_y-h(data_x,w,b)), data_x)*1/len(data_y)
  b =b + alpha*gradient_b
  w =w + alpha*gradient_w

home_25 = h(np.array([[8.3607,1.5,9.1500,1.777,2.0,8,4,48,1,1,38.9]]),w,b)
home_26 = h(np.array([[8.1400,1.0,8.0000,1.504,2.0,7,3,3,1,3,36.9]]),w,b)
home_27 = h(np.array([[9.1416,1.5,7.3262,1.831,1.5,8,4,31,4,1,45.8]]),w,b)
home_28 = h(np.array([[12.0000,1.5,5.0000,1.200,2.0,6,3,30,3,1,41.0]]),w,b)
```



```
print("home 25 (1): ", home_25 )
print("home 26 (0): ", home_26 )
print("home 27 (0): ", home_27 )
print("home 28 (1): ", home_28 )
```



```
[Running] python -u "c:\Users\user\Desktop\scienceComputer\year2\semesterB\4 מטלה\נים
home 25 (1):   [0.27752828]
home 26 (0):   [0.00044716]
home 27 (0):   [0.16413309]
home 28 (1):   [0.74062577]
```

confusion Matrix	classified as positive	classified as negative
really positive	1	1
really negative	0	2

Accuracy = (1+2)/1+1+2+0 = 0.75
Recall = 1 /1+1 = 0.5
Precision =1 / 1+0 = 1
F - measure =(2*1*0.5) / (0.5+1) = 2/3