Name: Tan Wen Tao Bryan

Class: DAAA/FT/2A/01

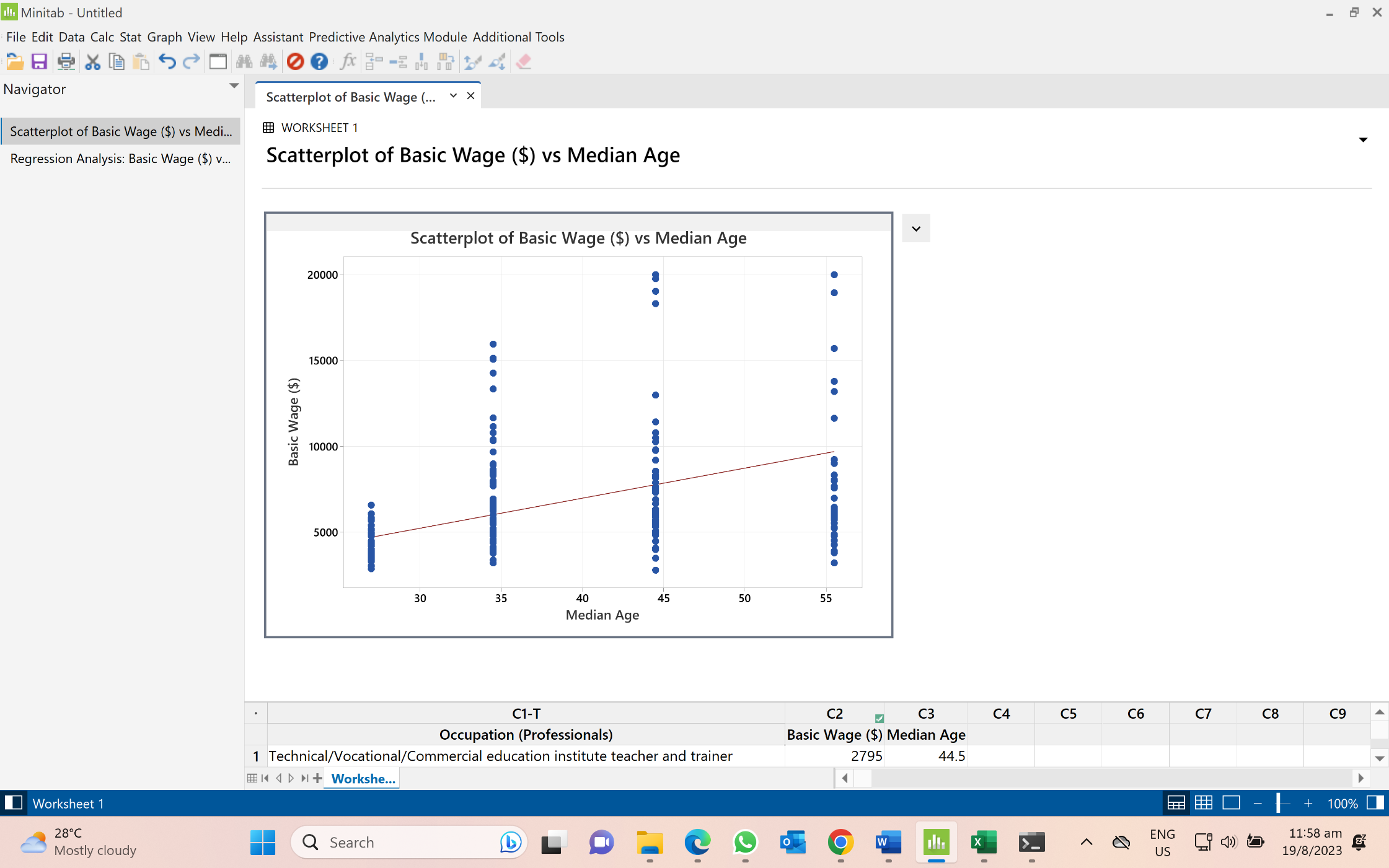
Student ID: 2214449

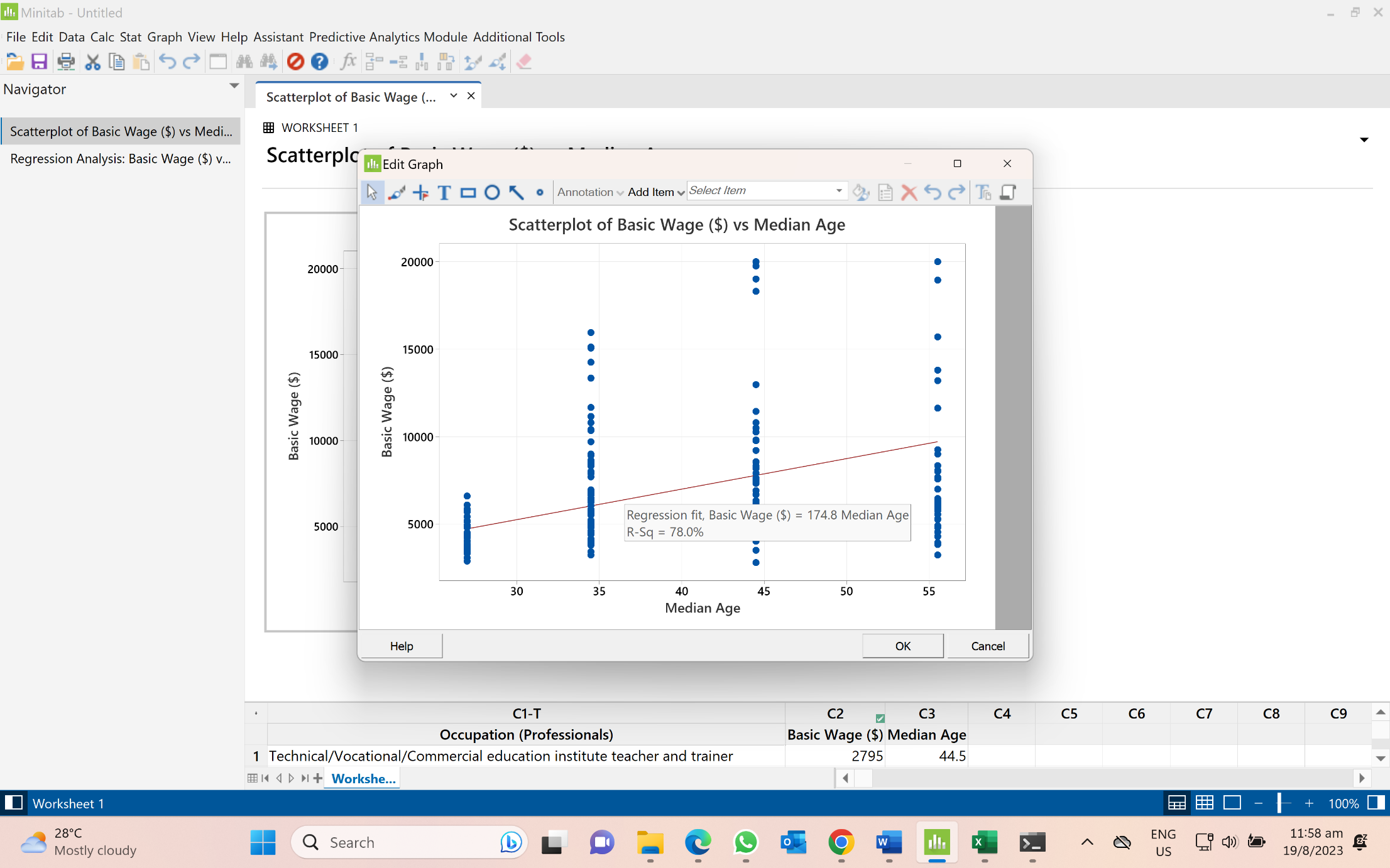
Dataset Used: 4. Wages 2021 June

**Qn 1:**

**a)**

**Scatterplot**





Equation of the line: = 174.8

**b)**

n = 232

**c)**

**i)**

**Code:**

b = 2 # Starting value of b

rate = 0.00001 # Set learning rate

precision = 0.001 # Stop algorithm when absolute difference between 2 consecutive x-values is less than precision

diff = 0.01 # difference between 2 consecutive iterates

max\_iter = 1000 # set maximum number of iterations

iter = 1 # iterations counter

#Create error function

def E\_fn1(b, x=data["Median Age"], y=data["Basic Wage ($)"]):

return sum((y-b\*x)\*\*2)/len(x)

#Create error derivative function

def derivE\_fn1(b, x=data["Median Age"], y=data["Basic Wage ($)"]):

return (2/len(x))\*sum((y-b\*x)\*(-x))

# Gradient Descent

while diff > precision and iter < max\_iter:

b\_new = b - rate \* derivE\_fn1(b)

print("Iteration ", iter, ": b-value is: ", b\_new,"f(b) is: ", E\_fn1(b\_new) )

diff = abs(b\_new - b)

iter = iter + 1

b = b\_new

print(f"Number of iterations is {iter-1}")

print(f"The local minimum occurs when b is {b}")

print(f"Minimum error is {E\_fn1(b)}")

**Output:**

Number of iterations is 264

The local minimum occurs when b is 174.7672096718554

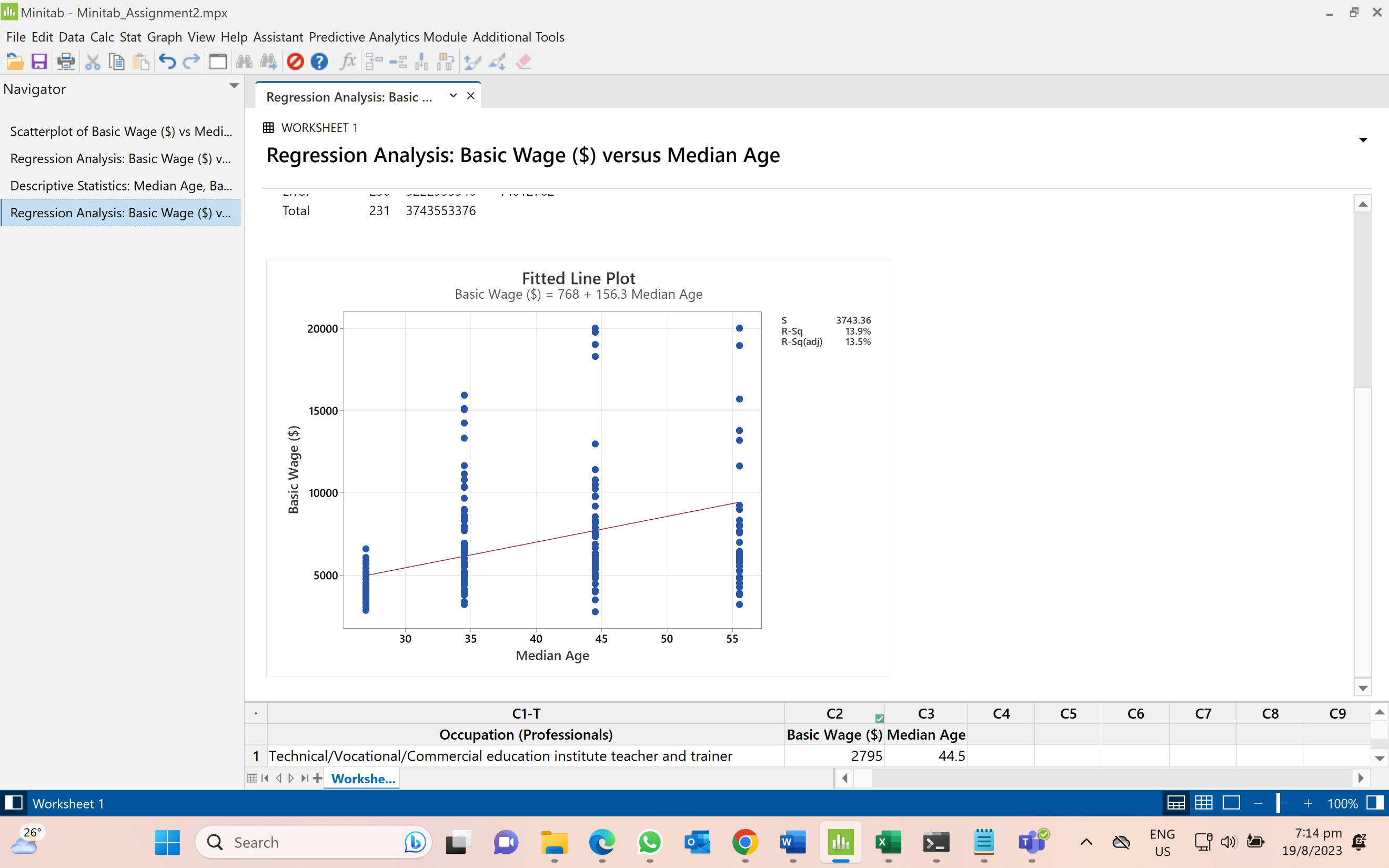
Minimum error is 13925466.39311236

**ii)**

Equation of the line: = 174.77

**Qn 2:**

**a)**

****

**b)**

**c)**

**i)**

**Code:**

next\_a = 6 # Initial point

next\_b = 10 # Initial point

alpha = 0.0005 # Learning rate

epsilon = 0.000005 # Stopping criterion constant

change\_func = 0.05

max\_iters = 150000 # Maximum number of iterations

iter = 1

#Create error function

def E\_fn2(a, b, x=data["Median Age"], y=data["Basic Wage ($)"]):

return sum((y-(a+b\*x))\*\*2)/len(x)

#Create error partial derivative function with respect to a

def partialDeriv\_a\_fn2(a, b, x=data["Median Age"], y=data["Basic Wage ($)"]):

return (-2/len(x))\*sum(y-(a+b\*x))

#Create error partial derivative function with respect to b

def partialDeriv\_b\_fn2(a, b, x=data["Median Age"], y=data["Basic Wage ($)"]):

return (2/len(x))\*sum((y-(a+b\*x))\*(-x))

next\_func = E\_fn2(next\_a,next\_b) # Initial value of function

while change\_func > epsilon and iter < max\_iters:

current\_a = next\_a

current\_b = next\_b

current\_func = next\_func

next\_a = current\_a-alpha\*partialDeriv\_a\_fn2(current\_a,current\_b) # update of a

next\_b = current\_b-alpha\*partialDeriv\_b\_fn2(current\_a,current\_b) # update of b

next\_func = E\_fn2(next\_a,next\_b)

change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge

print("Iteration",iter,": a = ",next\_a,", b = ",next\_b,", f(a,b) = ",next\_func, ", change\_func = ",round(change\_func,5))

iter+=1

print()

print(f"Number of iterations is {iter-1}")

print(f"The local minimum occurs when a is {next\_a} & b is {next\_b}")

print(f"Minimum error is {E\_fn2(next\_a, next\_b)}")

**Output:**

Number of iterations is 118903

The local minimum occurs when a is 766.6823731849599 & b is 156.27656031943158

Minimum error is 13891962.74248049

**ii)**

Equation of the line: = 766.68 + 156.28

**d)**

First, I decreased my learning rate to 0.0005 as a learning rate that is larger than this, minimum will be missed and the algorithm is unable to converge. If the learning rate is any smaller than this, it requires more than 150K iterations and the convergence is slow.

Secondly, I lowered my epsilon to 0.000005 such that convergence will be reach at a much more accurate point when the difference between is smaller than to show very little reduction between the two points.

Lastly, I have increased the maximum number of iterations to 150000 such that the algorithm is able to find the minimum error precisely and accurately when convergence is reached.

**Qn 3a)**

**Table: (Double Click to see All Records)**



**Table: (Double Click to see sample of 10 records used for Qn 3)**

****

New column I have added is **Average Monthly Working Hours** which is different for each occupation industry and can also contribute to the basic wage the occupations earn monthly.

Data Collection: I retrieved a text file which contains each individual **occupation** and what **industry sector** they belong to. I imported it to python and added a new column called **Average Monthly Working Hours** based on which **industry sector** **the occupation** belongs to. Afterwards, I merged this data frame and the original data frame together using **left join** to get a new data frame with the updated records as seen above. Lastly, I used pd.sample to get a **sample of 10 rows** from the dataset for the test.

Reference: <https://blog.seedly.sg/longest-working-hours-industries-jobs-singapore/>

**3b)**

**3c)**

**Code:**

next\_a = -57000 # Initial point

next\_b = 100 # Initial point

next\_c = 350 #Initial point

alpha = 0.00001 # Learning rate

epsilon = 0.0000001 # Stopping criterion constant

change\_func = 0.0001

max\_iters = 10000 # Maximum number of iterations

iter = 1

#Create error function

def E\_fn3(a, b, c, w=qn3\_df["Average Monthly Working Hours"], x=qn3\_df["Median Age"], y=qn3\_df["Basic Wage ($)"]):

return sum((y-(a+b\*x+c\*w))\*\*2)/len(x)

#Create error partial derivative function with respect to a

def partialDeriv\_a\_fn3(

a, b, c, w=qn3\_df["Average Monthly Working Hours"], x=qn3\_df["Median Age"], y=qn3\_df["Basic Wage ($)"]):

return (-2/len(x))\*sum(y-(a+b\*x+c\*w))

#Create error partial derivative function with respect to b

def partialDeriv\_b\_fn3(

a, b, c, w=qn3\_df["Average Monthly Working Hours"], x=qn3\_df["Median Age"], y=qn3\_df["Basic Wage ($)"]):

return (2/len(x))\*sum((y-(a+b\*x+c\*w))\*(-x))

#Create error partial derivative function with respect to c

def partialDeriv\_c\_fn3(

a, b, c, w=qn3\_df["Average Monthly Working Hours"], x=qn3\_df["Median Age"], y=qn3\_df["Basic Wage ($)"]):

return (2/len(x))\*sum((y-(a+b\*x+c\*w))\*(-w))

next\_func = E\_fn3(next\_a,next\_b, next\_c) # Initial value of function

while change\_func > epsilon and iter < max\_iters:

current\_a = next\_a

current\_b = next\_b

current\_c = next\_c

current\_func = next\_func

next\_a = current\_a-alpha\*partialDeriv\_a\_fn3(current\_a,current\_b,current\_c) # update of a

next\_b = current\_b-alpha\*partialDeriv\_b\_fn3(current\_a,current\_b,current\_c) # update of b

next\_c = current\_c-alpha\*partialDeriv\_c\_fn3(current\_a,current\_b,current\_c) # update of c

next\_func = E\_fn3(next\_a,next\_b,next\_c)

change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge

print("Iteration",iter,": a = ",next\_a,", b = ",next\_b,", c = ",next\_c,", f(a,b,c) = ",next\_func,", change\_func = ",round(change\_func,5))

iter+=1

print()

print(f"Number of iterations is {iter-1}")

print(f"The local minimum occurs when a is {next\_a}, b is {next\_b} & c is {next\_c}")

print(f"Minimum error is {E\_fn3(next\_a, next\_b, next\_c)}")

**Output:**

Number of iterations is 5473

The local minimum occurs when a is -59345.92662933647, b is 104.88740243282223 & c is 373.0511097800096

Minimum error is 3045003.0860464624

**Equation:**

= -5934.93 + 104.89 + 373.05

**Minitab Analysis:**

