**NUMPY**

* Numpy is an open source Python library which consists of mathematical functions for data manipulation.
* Firstly, to use the functionalities of Numpy, we need to install and import the library in whichever environment we are working on (Jupyter Notebook, Google Colab etc.).

!pip install numpy

import numpy as np

or

import numpy

*Note:* ‘np’ is just an alias used as a convention in the Industry for ease of writing code. You can use any alias to represent it.

* **Arrays and Matrices:**

We can create arrays or N Dimensional matrices

arr =np.array ([2,3], [4,5])

This gives us a 2x2 matrix

We can also create matrices of mxn dimensions with the following values:

np.zeros((m,n)) consists of zeros.

np.ones((m,n)) consists of ones.

np.full((m,n), constant) consists of the constant value in every position.

np.eye(m) is an identity matrix and a square matrix(mxm).

* **Basic Mathematical operations:**

If arr1 and arr2 are two arrays,

np.add(arr1 +arr2) returns their sum

Similarly, we have

np.subtract(arr1+arr2) for their difference

np.dot(arr1, arr2) for their product

We can find the sum of a matrix x by

np.sum(x)

or reshape a matrix x using

x.reshape((m,n))

np.unique(arr, return\_count=True) returns the elements occurring in the array without repetition and their frequency.

* **Statistical Functions:**

*Note*: x can be a matrix or an array.

np.argmin(x) returns the index at which the minimum value is present in array arr.

np.max(x) and np.min(x) return the maximum and minimum values present, respectively.

We can add the axis=0 or 1 parameter to find the min and max values along the rows or columns in a matrix.

np.mean(x)

np.median(x)

np.mode(x)

np.average(x)

np.std(x)

np.var(x)

return important statistics like the mean, median,mode, average, standard deviation and variance values in floating point.

* **Random Module:**

This special module in the Numpy library has numerous functions which allow us to generate random values.

Firstly, we import the library

import np.random

1. rand() – Returns random values in a given shape

eg: np.random.rand(2,3) returns a matrix of shape 2x3 with random values.

1. randn() – Returns a sample s from the standard normal distribution.
2. randint(low, high, n)- Returns n random integers from the range low(inclusive) to high(exclusive).

eg: np.random.int(1,28,3) returns 3 random integers between 1 and 28

1. random()- Returns random floats in the half-open interval [0.0, 0.1)
2. choice()- Generates a random sample from a given 1-D array.

eg: np.random.choice([1,3,4,5,6])

1. shuffle()- Shuffles the contents of a given sequence

eg: np.random.shuffle(arr)

**PANDAS**

* Pandas is an open source Python library used for data analysis and manipulation in Python. It is an extension of the Numpy Library.
* Pandas is used for reading (or loading) and writing data between in-memory data structures and files. E.g.: Comma Separated Files(.csv), SQL Databases, Excel sheets(.xlsx).
* Other functions include reshaping, slicing, indexing, merging and joining datasets.
* The core component of Pandas Library is a *Data Frame* which is a special object which stores the data in tabular form (rows and columns).

Let’s go over the functions and their syntax provided by this library. We will use a sample dataset ‘diabetes.csv’ throughout.

* Firstly, to use the functionalities of Pandas, we need to install and import the library in whichever environment we are working on (Jupyter Notebook, Google Colab etc.).

!pip install pandas

import pandas as pd

or

import pandas

*Note:* ‘pd’ is just an alias used as a convention in the Industry for ease of writing code. You can use any alias to represent it.

* **Input/ Output functions** i.e. Reading and Writing files.

Loading/ Reading data:

pandas.read\_csv(‘diabetes.csv’)

*Note:* We can read other file formats using the syntax read\_fileformat. Eg: read\_json, read\_excel, etc.

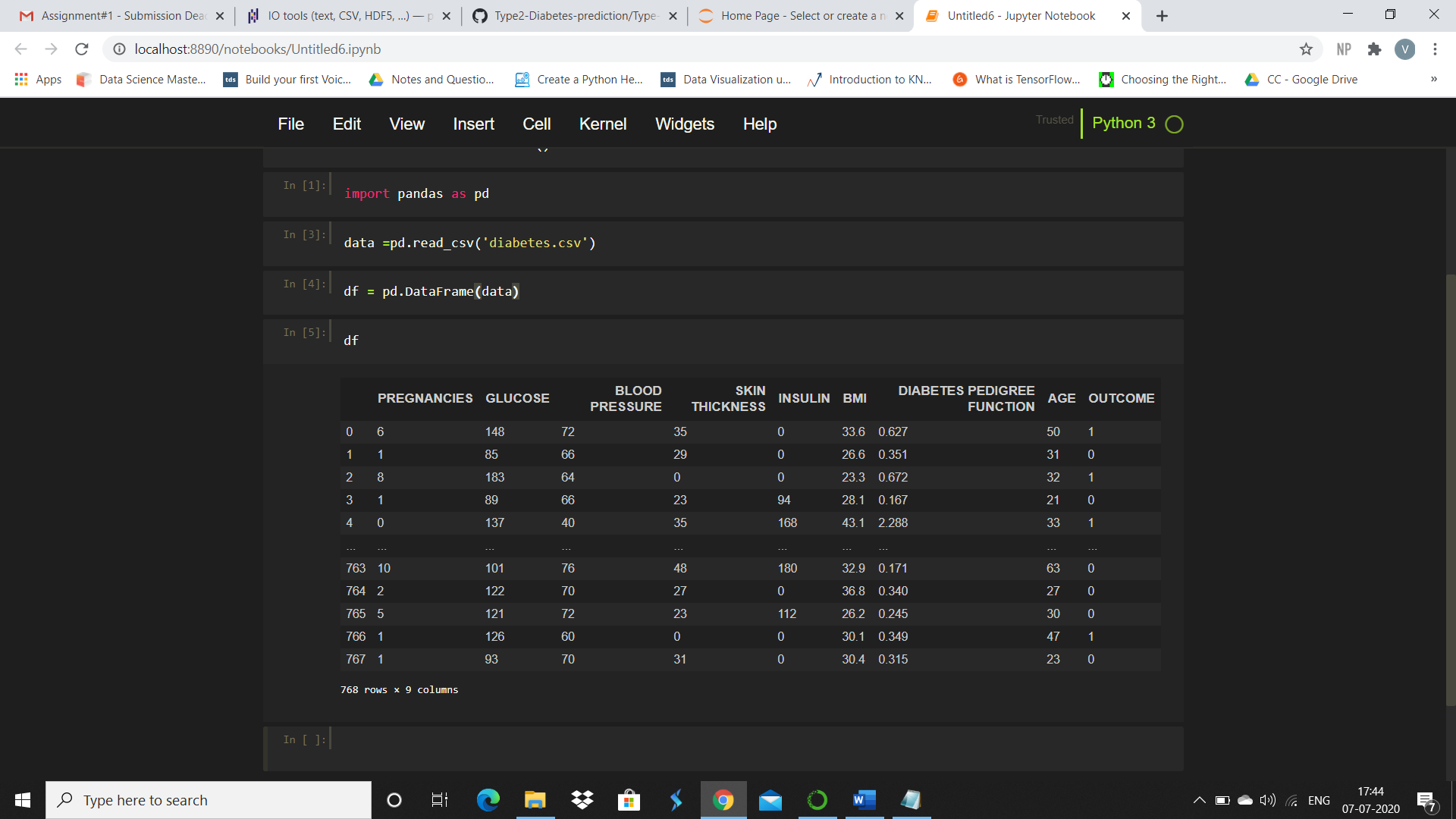
We can even load data into a data frame to view it in a tabular form if it is in a different format like a dictionary.

From pandas import DataFrame

data = pandas.read\_csv(‘diabetes.csv’)

df = pandas.DataFrame(data)

df



The line 768 rows x 9 columns represent the dimensions of our dataset.

*Note:* The User can use any alias instead of ‘data’ and ‘df’.

Thereafter, we can convert the data to another file format and save it.

df.to\_csv(‘diabetes\_new.csv’)

* **Displaying and Learning about data**

We can view the first n rows of the dataset

df.head(n)

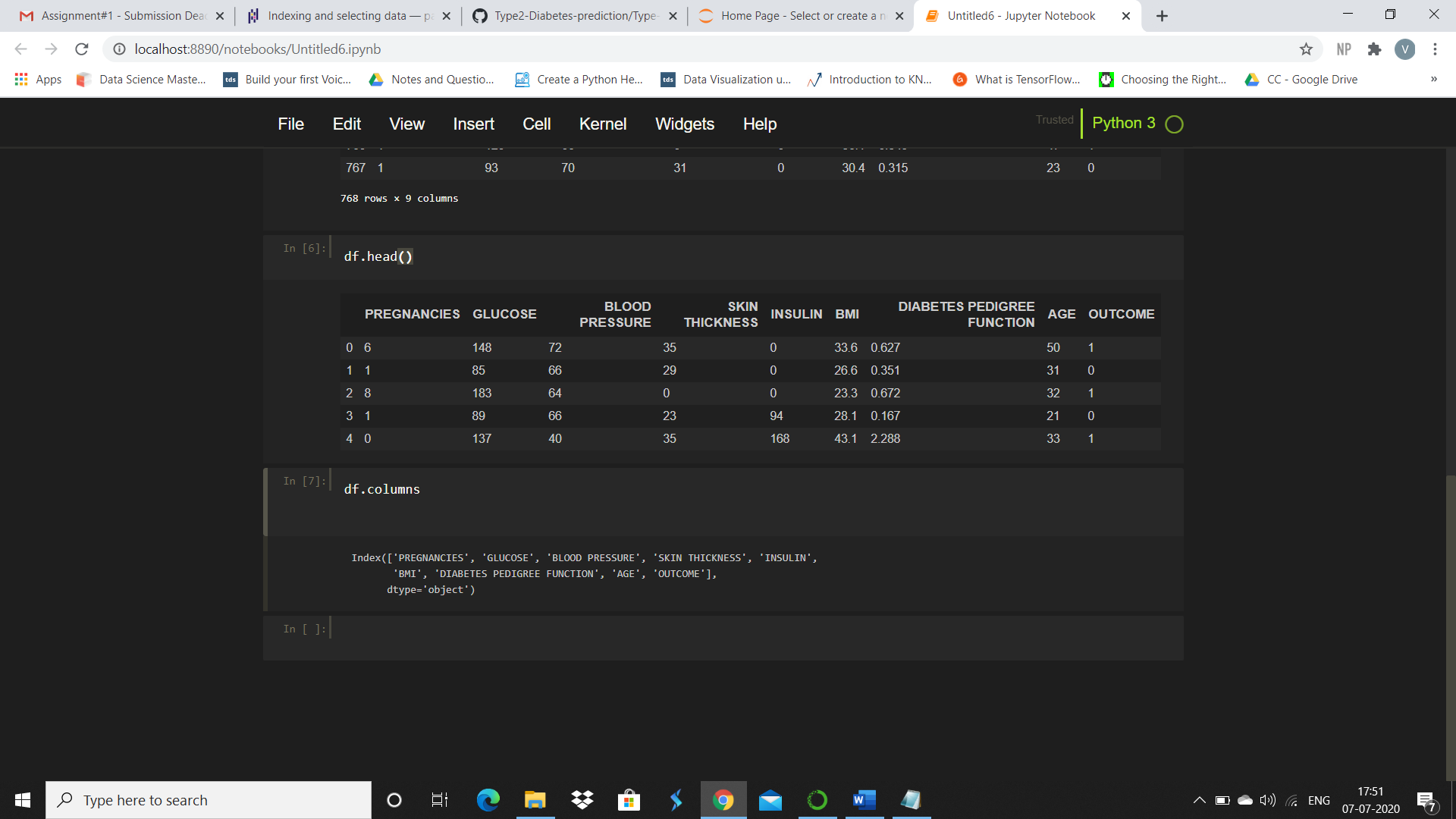
Or the last n rows

df.tail(n)

*Note:* The default value of n = 5.

We can view the column titles/headings and the data types of their values.

df. columns

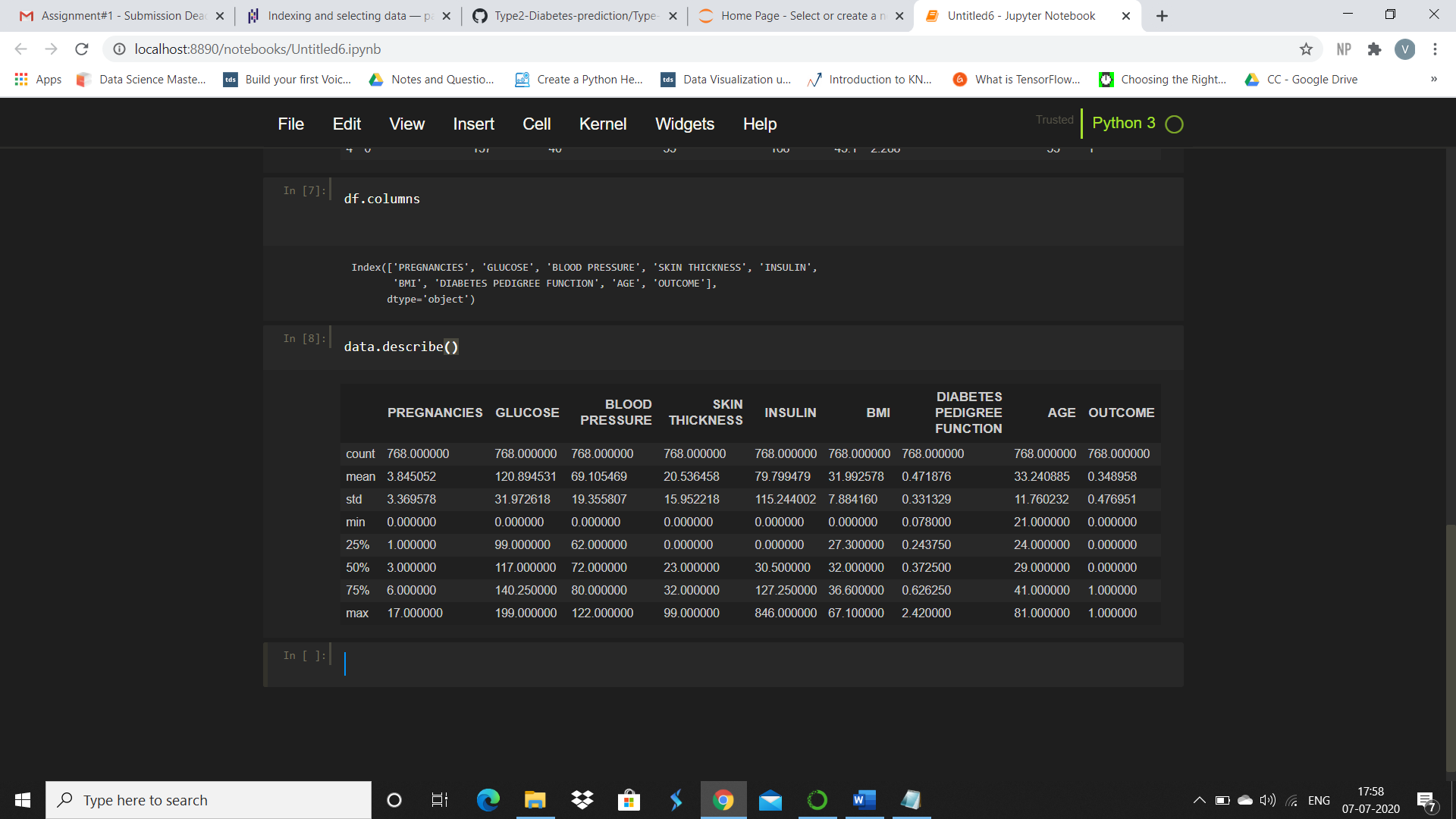


Or any number of specific columns

df[[‘Columnname1’, ‘Columnname2’]]

We can also display certain statistics related to our dataset.

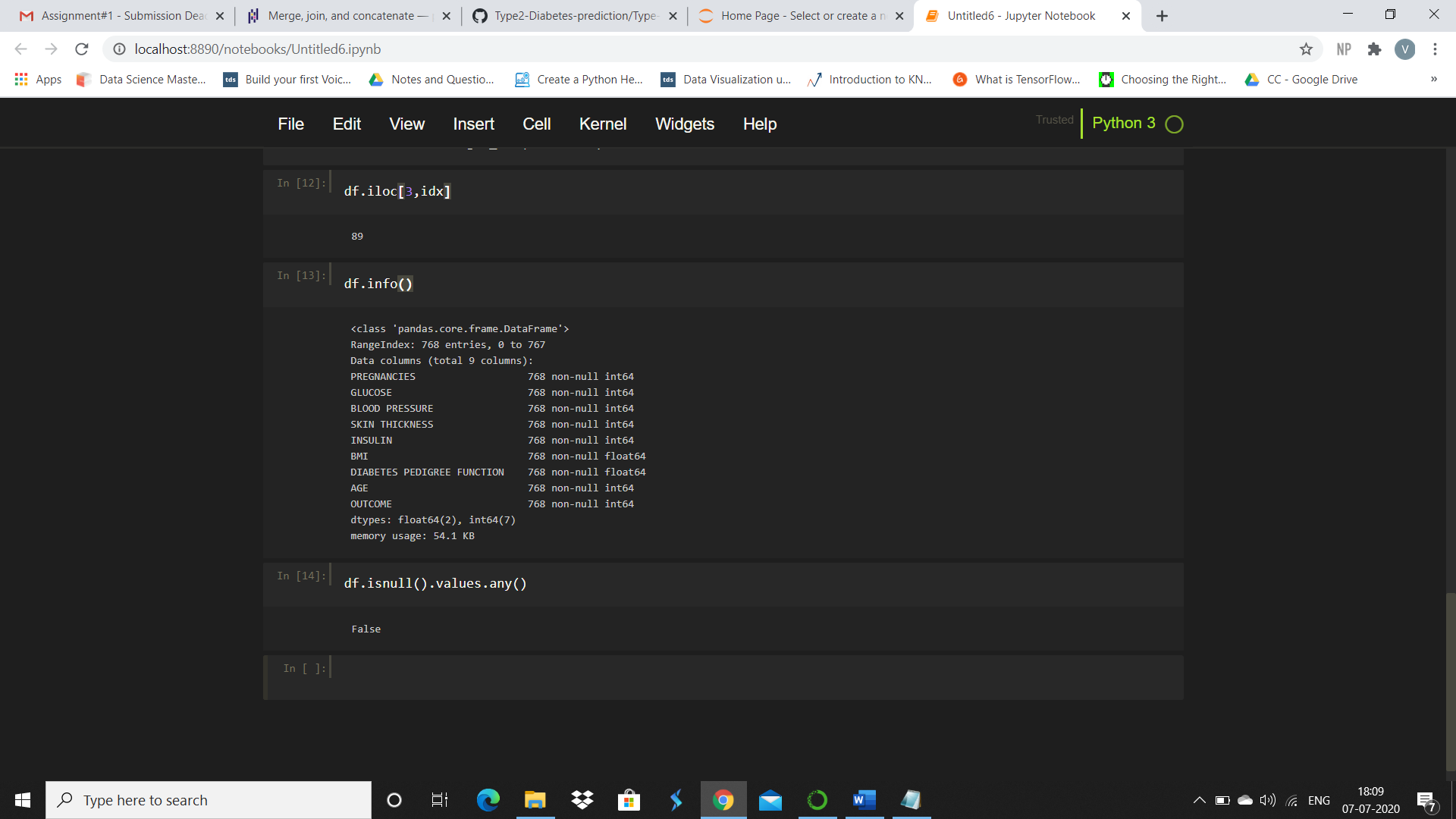
df.describe()



A similar function gives us the information on the data types of each column and the number of non-null values.

df.info()

We can also use df.isnull().values.any() to check for any null values

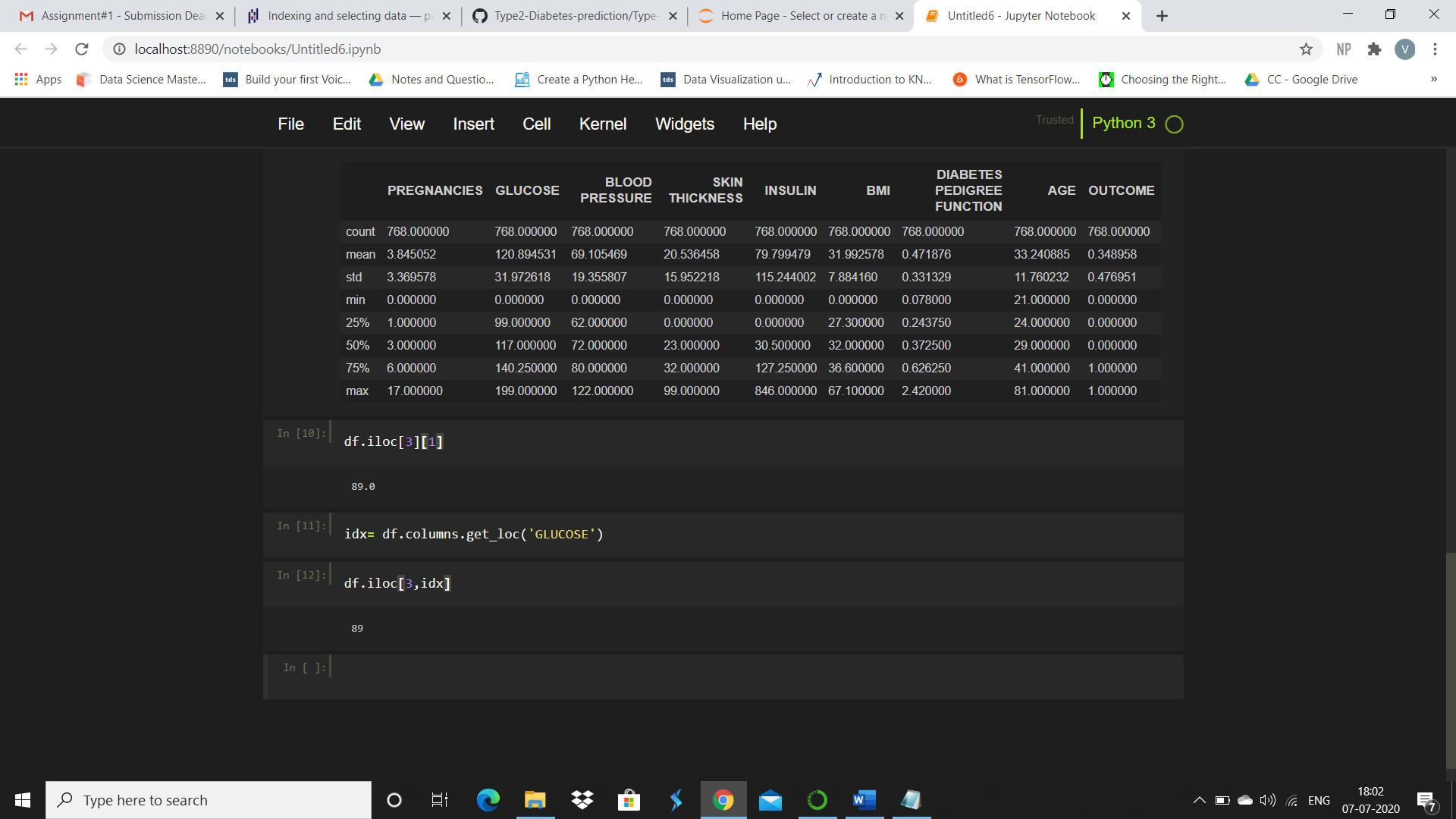


We can also access a particular value by defining the row and column values

df.iloc[row\_value][column\_value]

or a particular row value from a column defined by its name.

eg:



* **Drop or delete a column/row**

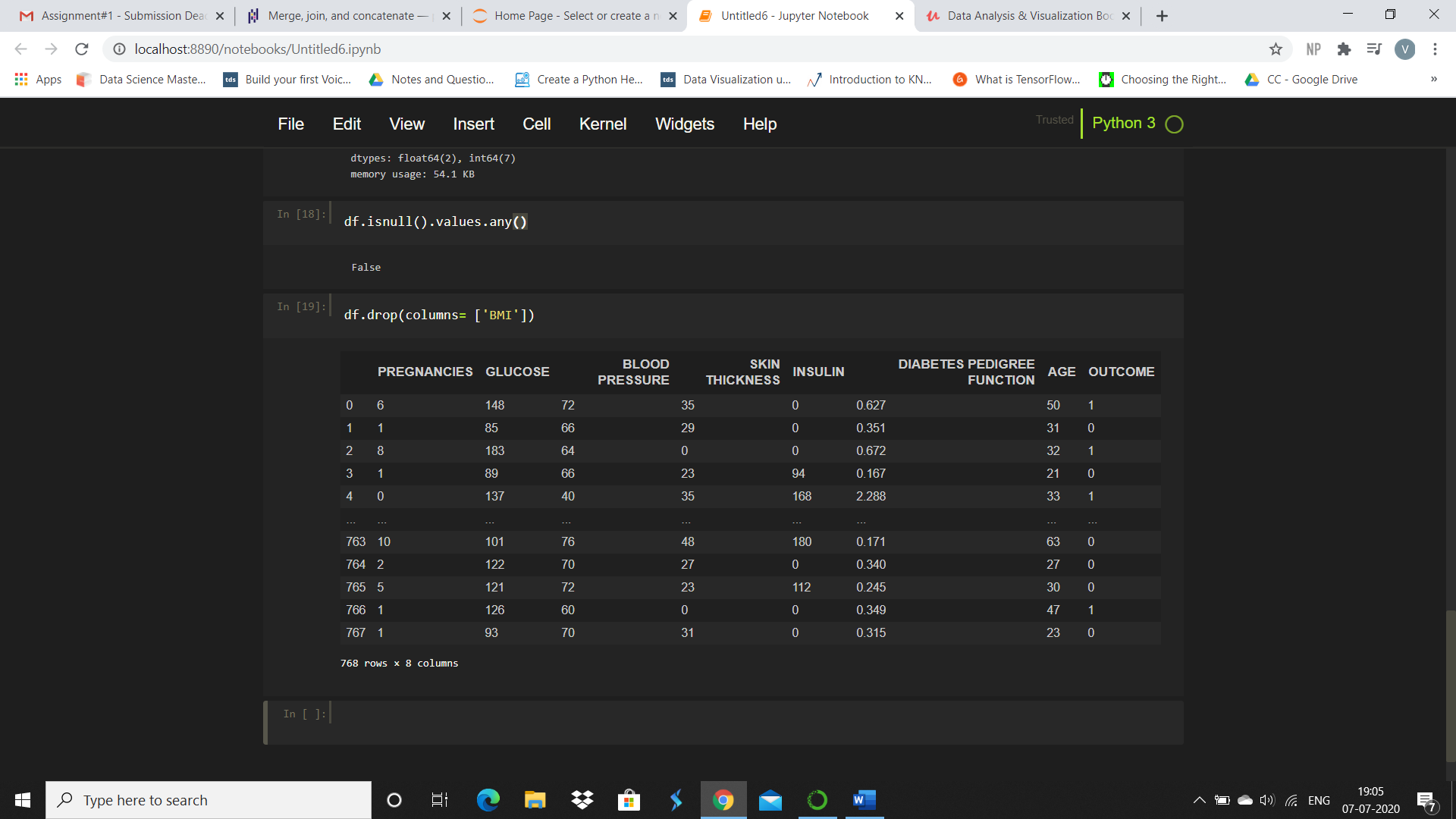
del(df[‘col\_name’])

or

df= df.drop(column=[‘ col\_name’])

or

df = df.drop(‘col\_name’, axis=1) to drop a column.



Similarly,

df = df.drop(‘row\_name’, axis=0) to drop a row.

We can also drop rows(axis=0) or columns(axis=1) with NAN values

df.dropna(axis=0 or 1 , how=’all’, thresh= n) where how is a parameter which drops the row/col if all values are NAN values and thresh is the minimum number of NAN values to drop the row/col.

We can fill the NAN values with 0

df.fillna(0)