

**ANL252**

**Python for Business Analytics**

# **End of Course Assignment**

**July 2021**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Tia Kang Jun** | **B2070385** |

**Tutorial Group: ­­­­­­­­­­­­ T 09**

**Instructor’s Name: Mr. Kumar Manish**

**Submission Date: 12/09/2021**

1ai)

#import numpy and pandas

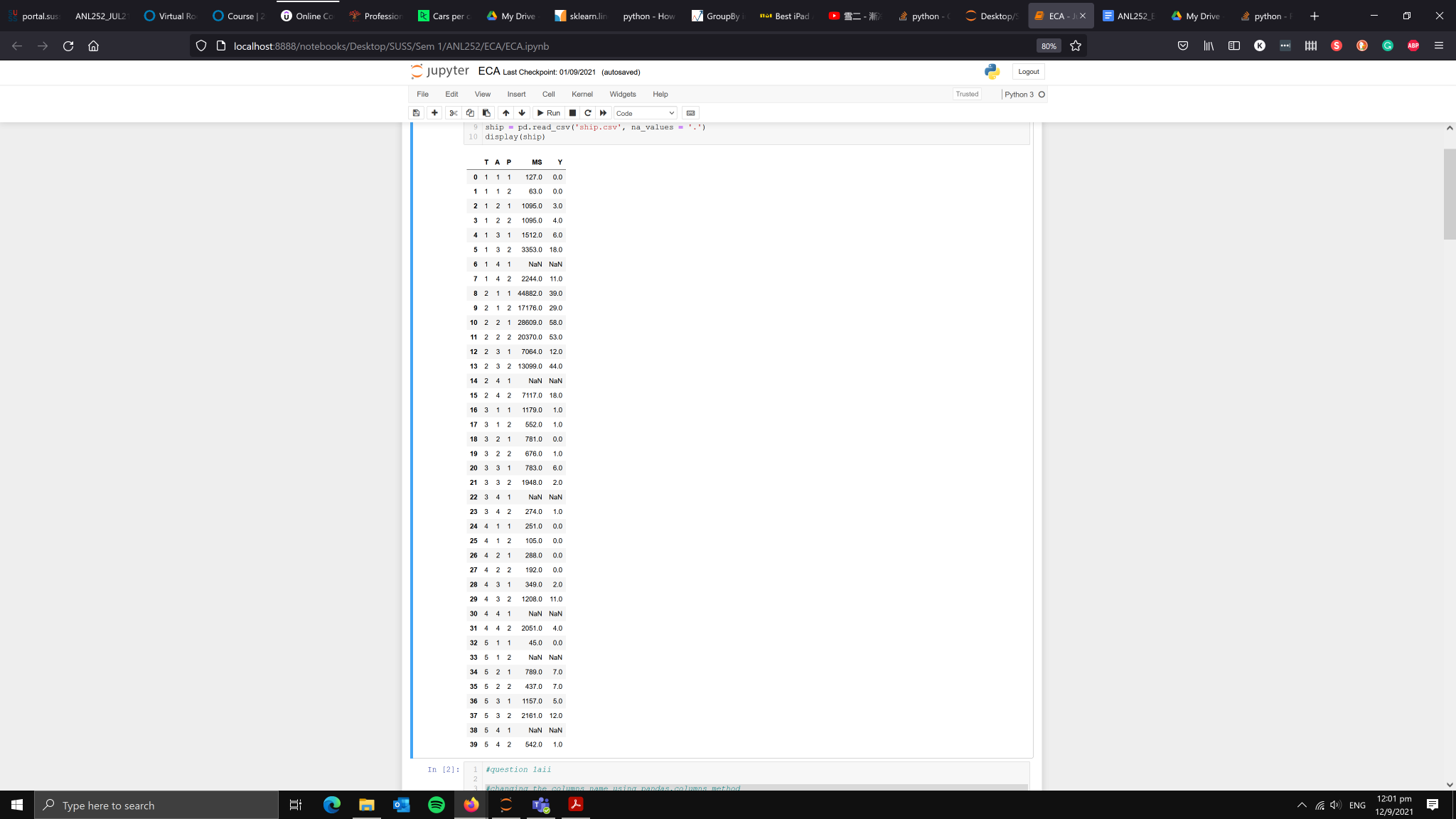
import numpy as np

import pandas as pd

#reading the csv file into python by using pandas.read\_csv method with . set as a missing value

ship = pd.read\_csv('ship.csv', na\_values = '.')

display(ship)

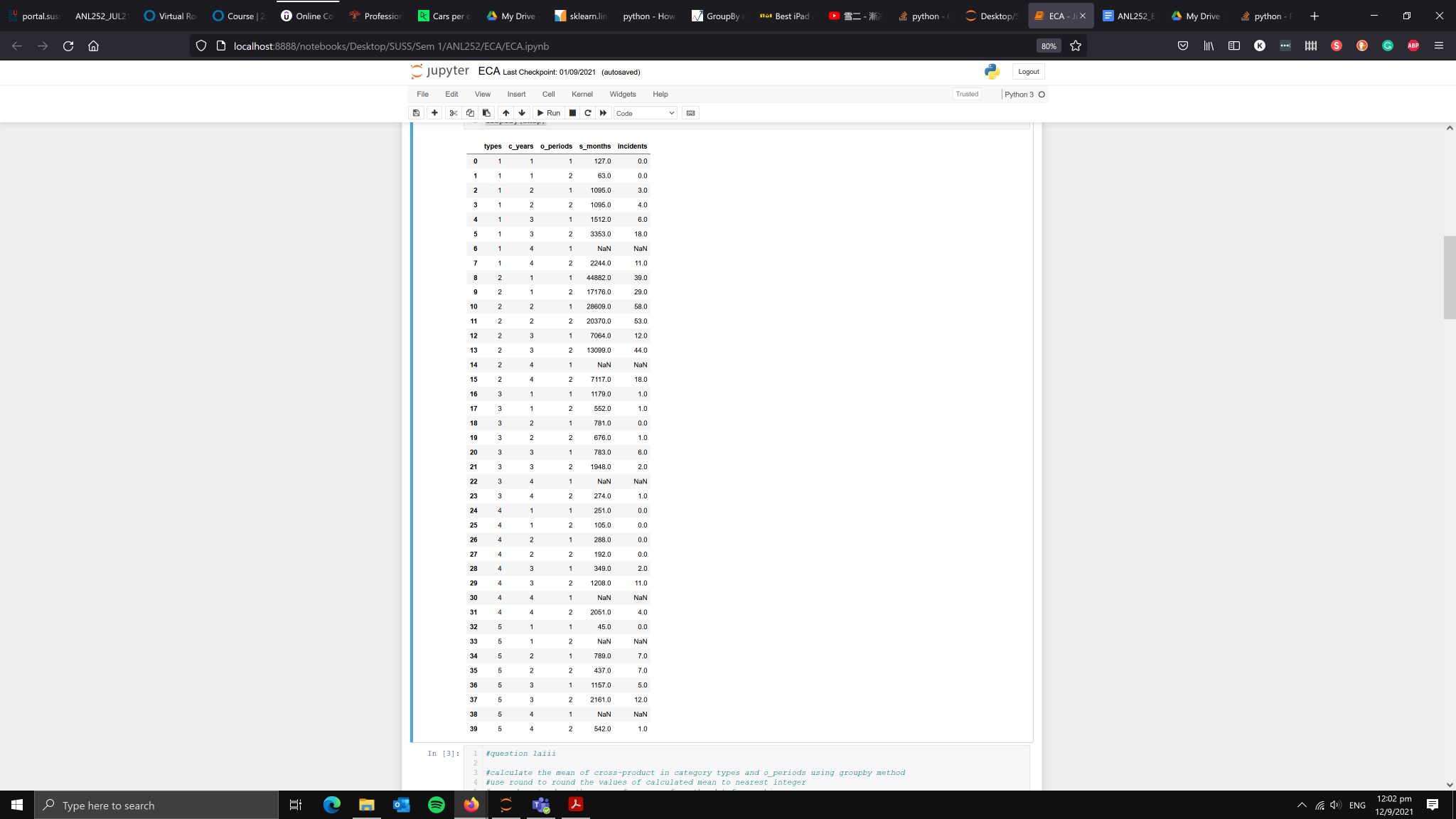


1aii)

#changing the columns name using pandas.columns method

ship.columns = ['types','c\_years','o\_periods','s\_months','incidents']

display(ship)



1aiii)

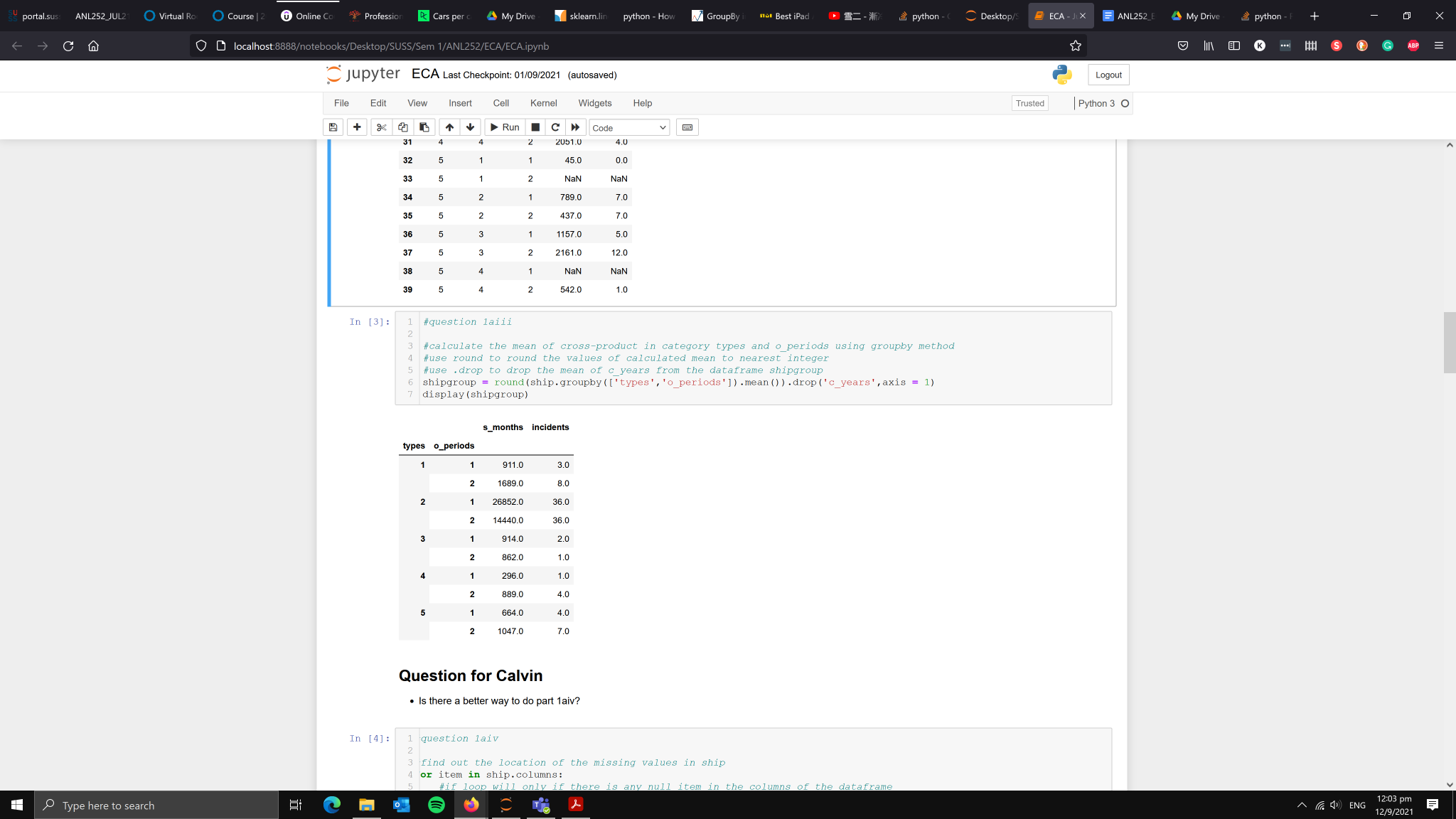
#calculate the mean of cross-product in category types and o\_periods using groupby method

#use round to round the values of calculated mean to nearest integer

#use .drop to drop the mean of c\_years from the dataframe shipgroup

shipgroup = round(ship.groupby(['types','o\_periods']).mean()).drop('c\_years',axis = 1)

display(shipgroup)



1aiv)

#find out the location of the missing values in ship

for item in ship.columns:

#if loop will only run if there is any null item in the columns of the dataframe

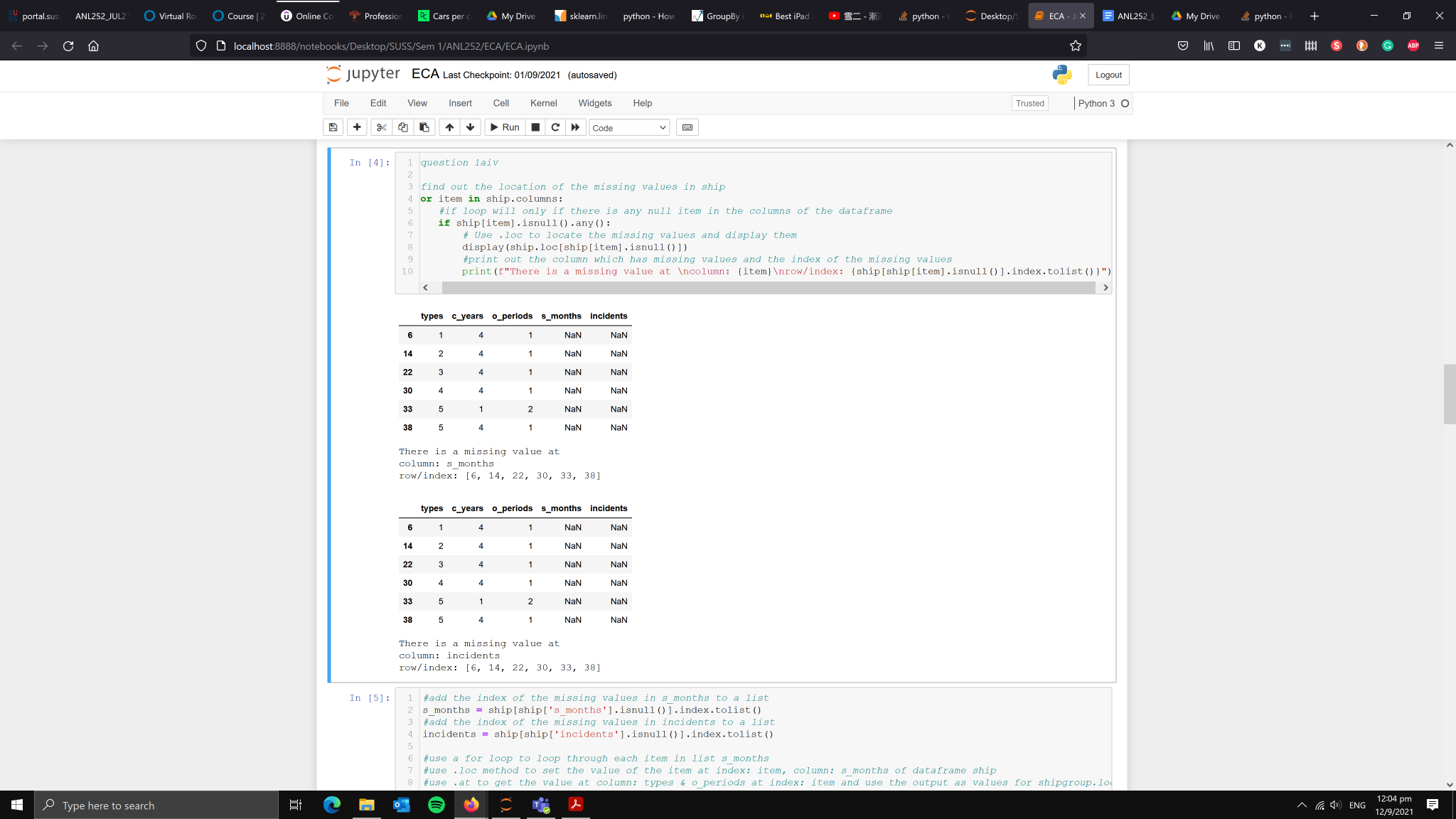
if ship[item].isnull().any():

# Use .loc to locate the missing values and display them

display(ship.loc[ship[item].isnull()])

#print out the column which has missing values and the index of the missing values

print(f"There is a missing value at \ncolumn: {item}\nrow/index: {ship[ship[item].isnull()].index.tolist()}")



#add the index of the missing values in s\_months to a list

s\_months = ship[ship['s\_months'].isnull()].index.tolist()

#add the index of the missing values in incidents to a list

incidents = ship[ship['incidents'].isnull()].index.tolist()

#use a for loop to loop through each item in list s\_months

#use .loc method to set the value of the item at index: item, column: s\_months of dataframe ship

#use .at to get the value at column: types & o\_periods at index: item and use the output as values for shipgroup.loc

#set the value of ship.loc to be value of shipgroup.loc

for item in s\_months:

ship.loc[item,'s\_months'] = shipgroup.loc[ship.at[item,'types'],ship.at[item,'o\_periods']]['s\_months']

#use a for loop to loop through each item in list incidents

#use .loc method to get the value of the item at index: item, column: incidents of dataframe ship

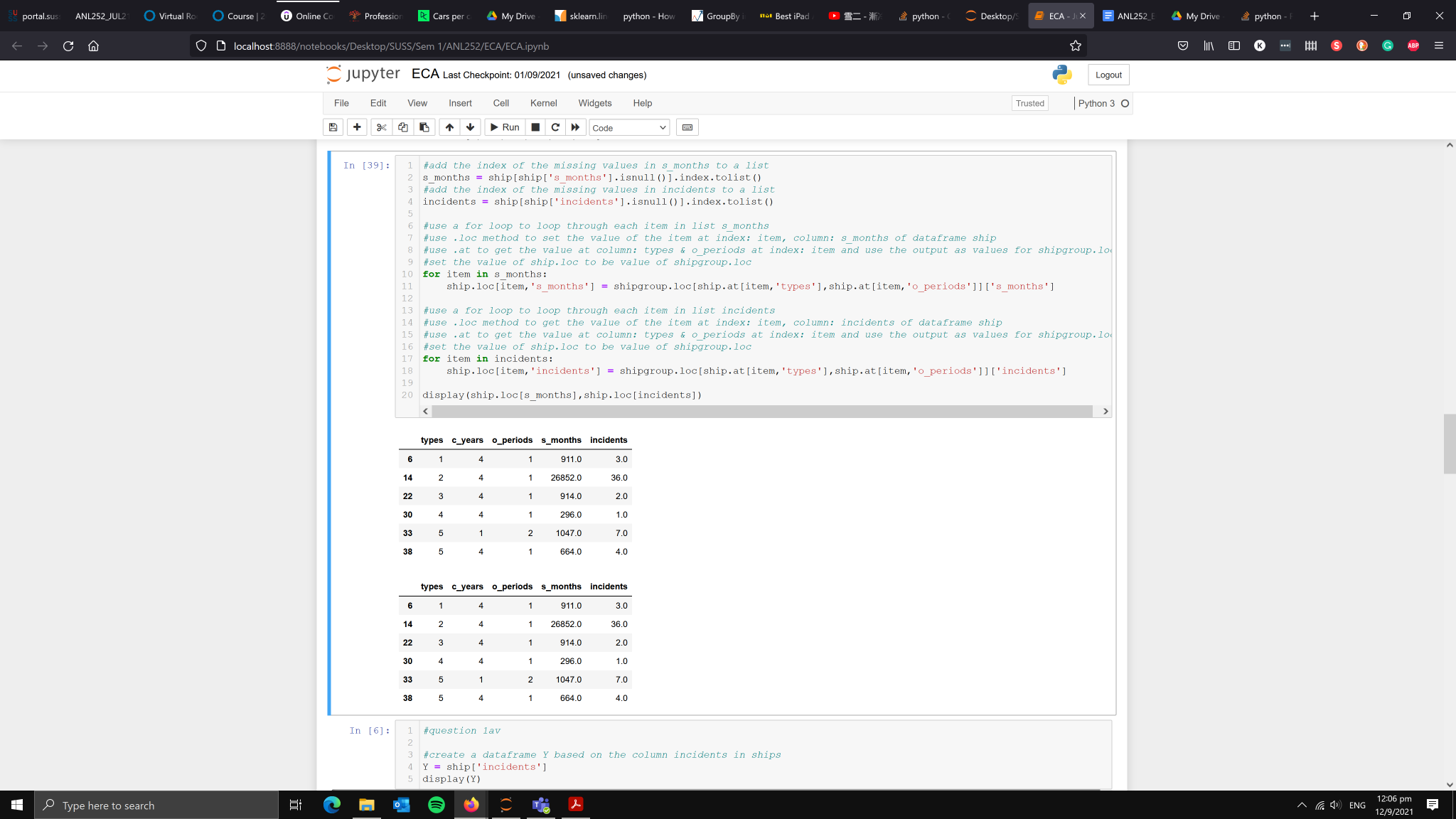
#use .at to get the value at column: types & o\_periods at index: item and use the output as values for shipgroup.loc

#set the value of ship.loc to be value of shipgroup.loc

for item in incidents:

ship.loc[item,'incidents'] = shipgroup.loc[ship.at[item,'types'],ship.at[item,'o\_periods']]['incidents']

display(ship.loc[s\_months], ship.loc[incidents])

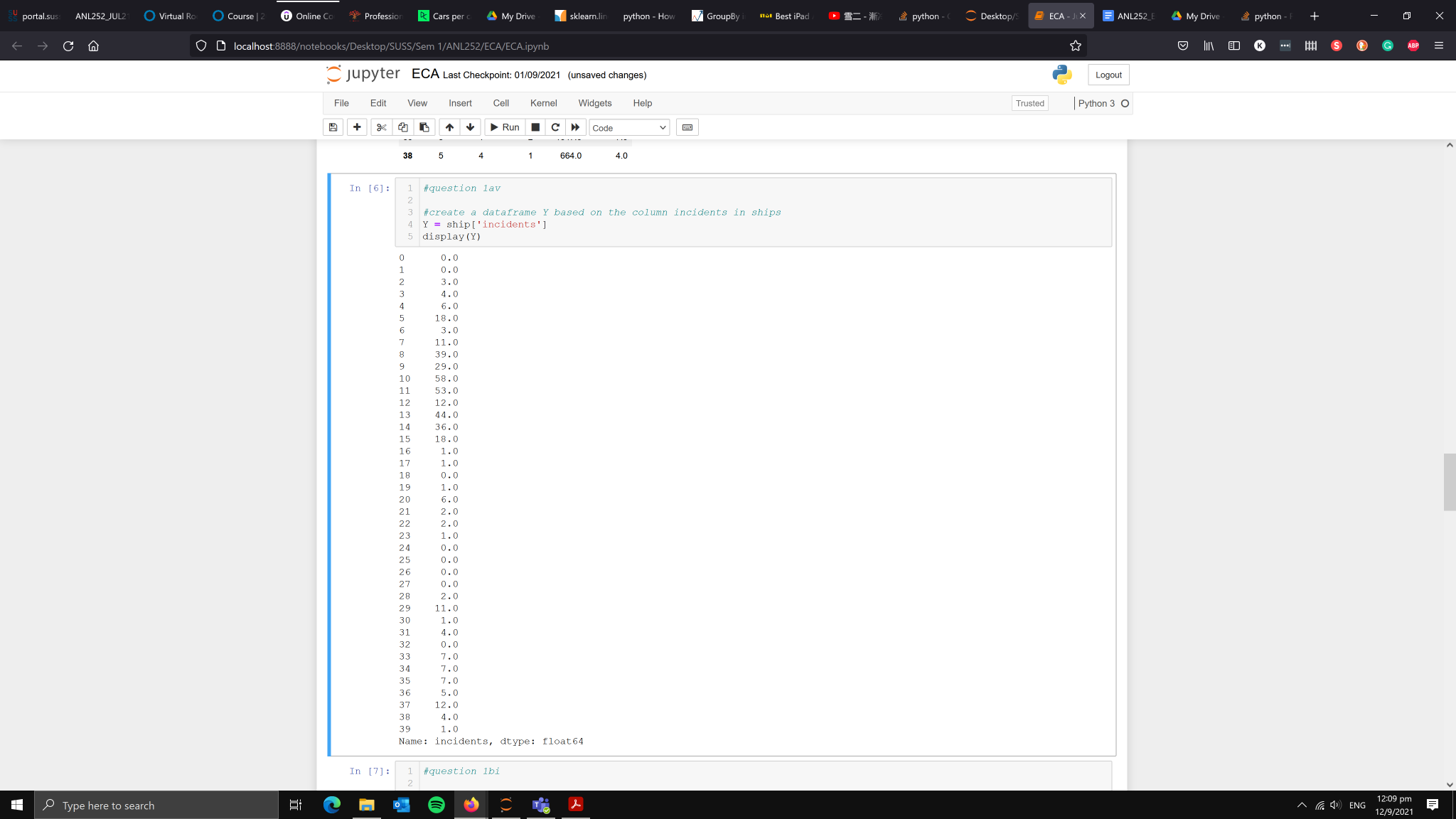


1av)

#create a dataframe Y based on the column incidents in ships

Y = ship['incidents']

display(Y)

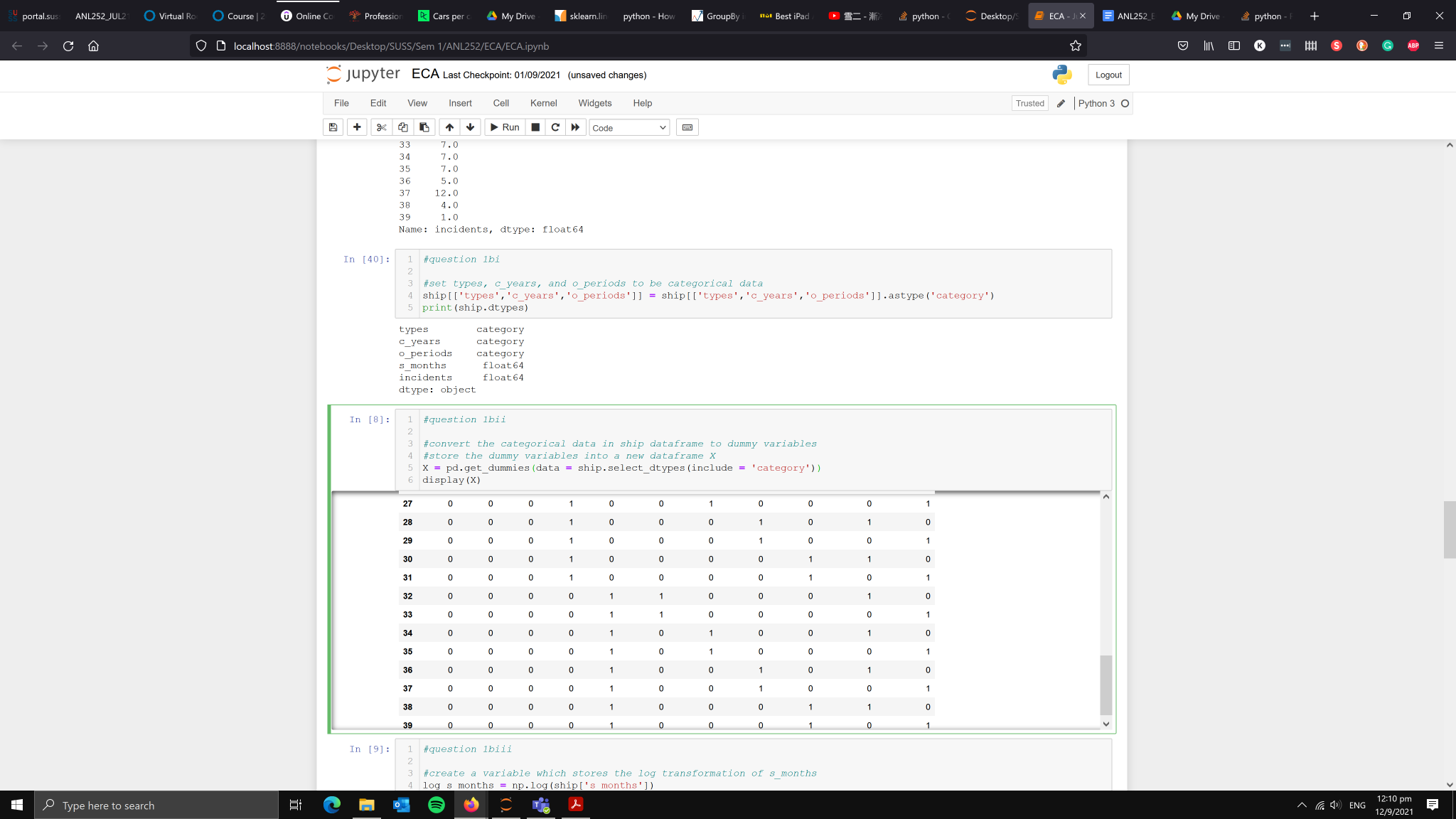


1bi)

#set types, c\_years, and o\_periods to be categorical data

ship[['types','c\_years','o\_periods']] = ship[['types','c\_years','o\_periods']].astype('category')

print(ship.dtypes)



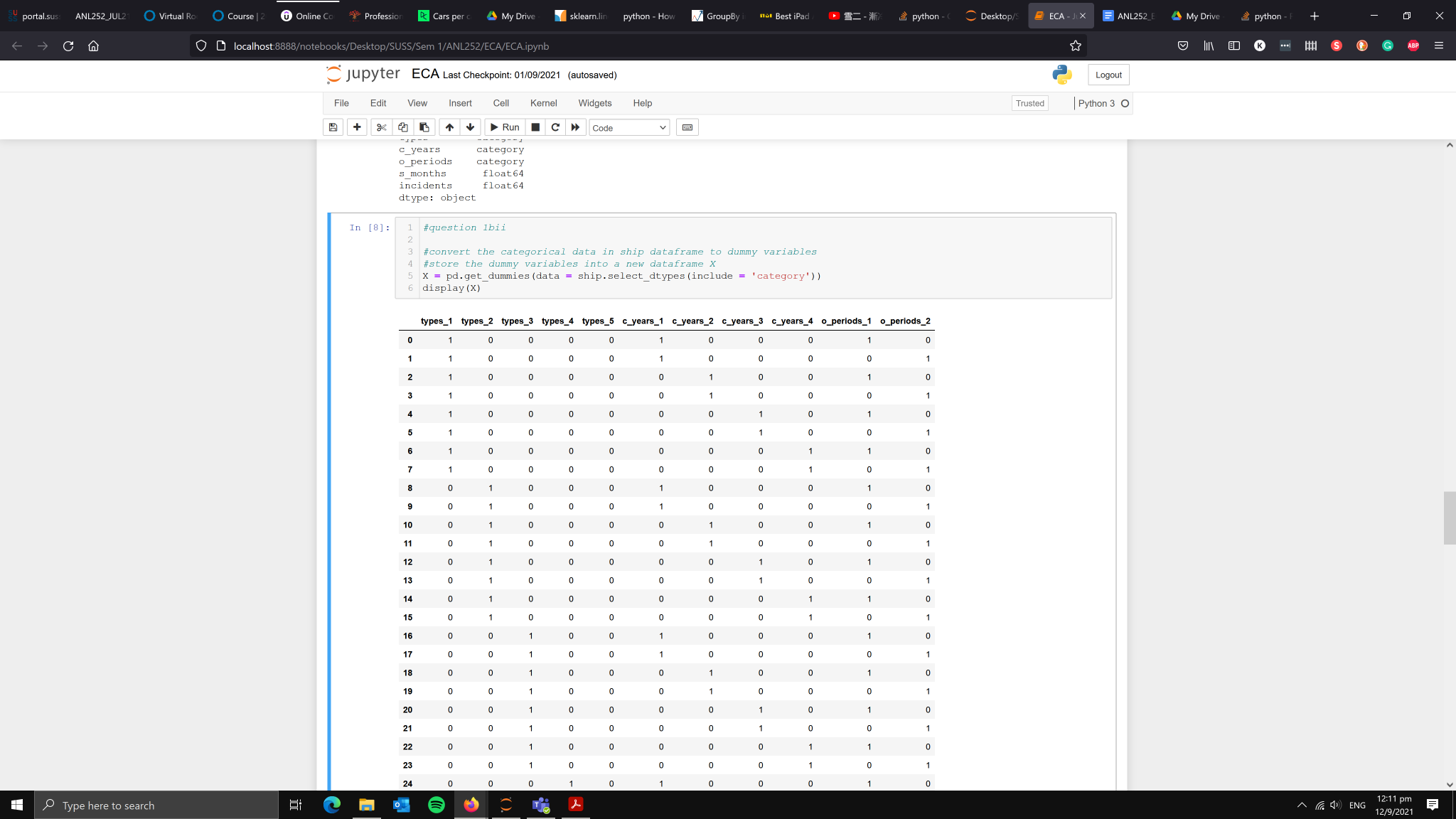
1bii)

#convert the categorical data in ship dataframe to dummy variables

#store the dummy variables into a new dataframe X

X = pd.get\_dummies(data = ship.select\_dtypes(include = 'category'))

display(X)



1biii)

#create a variable which stores the log transformation of s\_months

log\_s\_months = np.log(ship['s\_months'])

#append a new column called log\_s\_months into dataframe ship based on the variable log\_s\_months

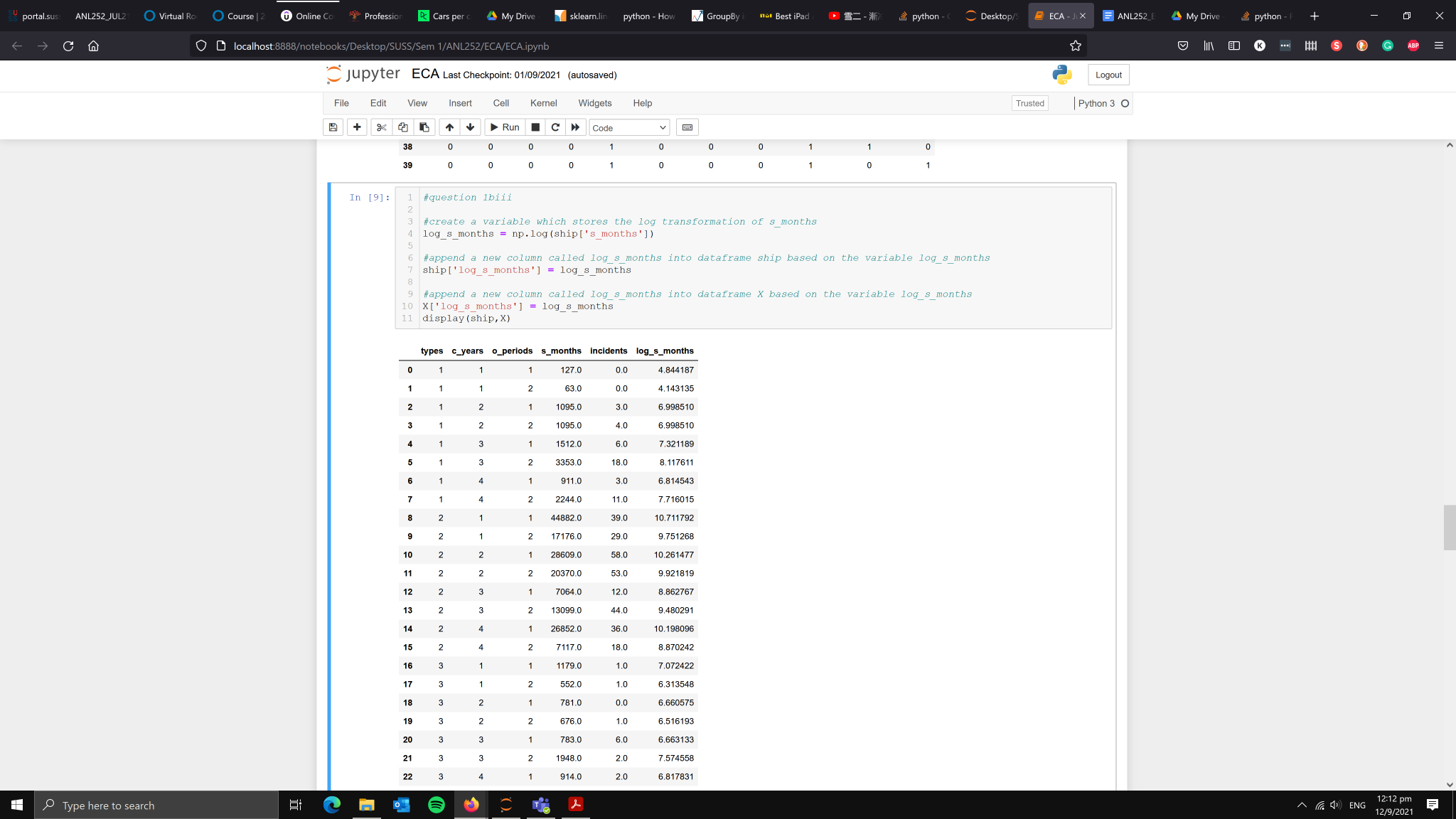
ship['log\_s\_months'] = log\_s\_months

#append a new column called log\_s\_months into dataframe X based on the variable log\_s\_months

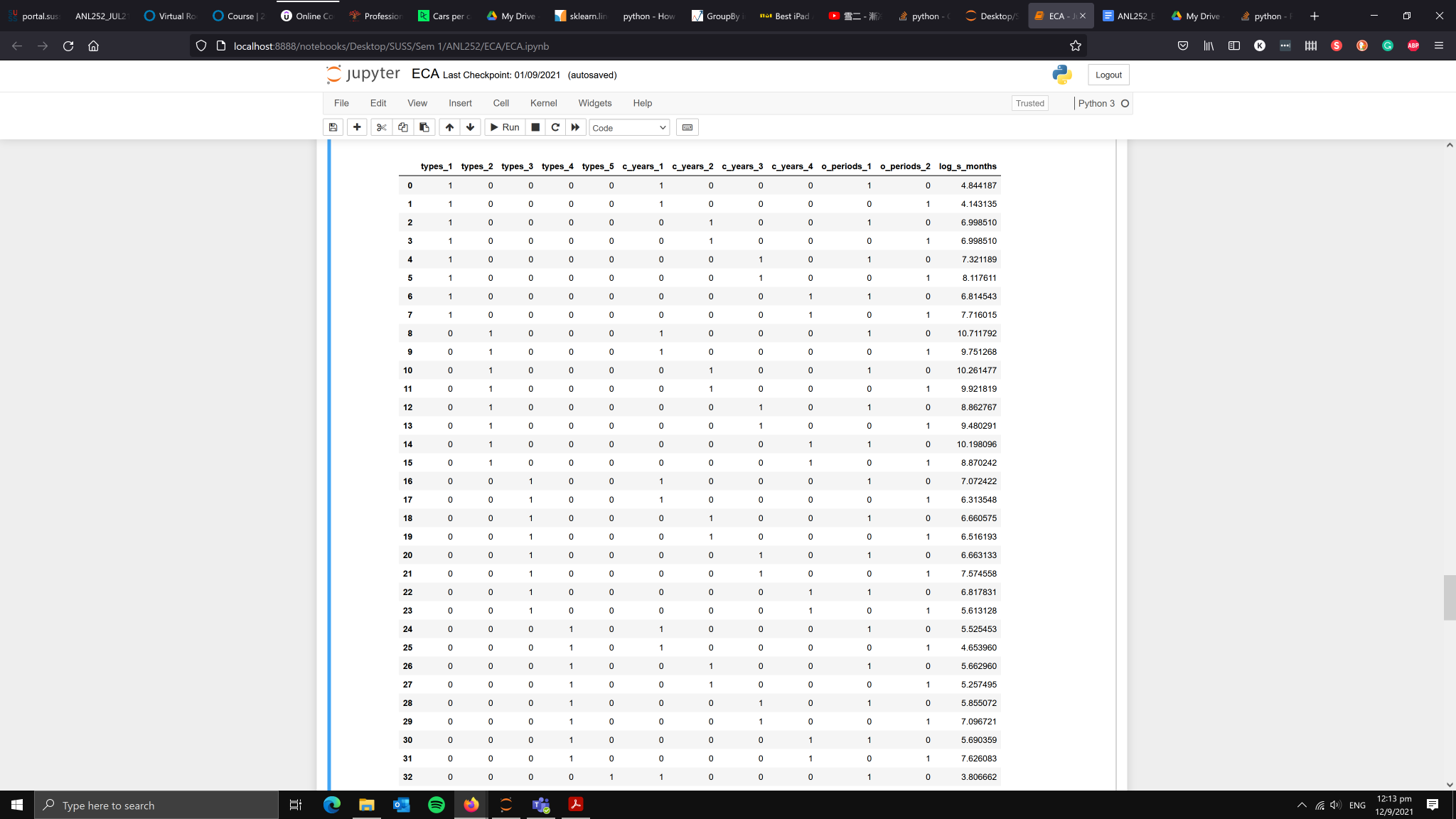
X['log\_s\_months'] = log\_s\_months

display(ship,X)

Ship dataframe:



X dataframe:



1c) After studying the dataframe, one of the main reasons why we do not split it to train and test datasets is because of the number of datapoints that we have. If we look at the ship dataframe, we can see that for each cross-product of ship types and operating periods we only have 4 datapoints to look at. By further splitting the model to a 75/25 train test split, we will only have 3 datapoints for the model to look at. This will cause the prediction to be largely inaccurate due to the lack of information for the model to learn from.

1d)

#import sqlite3 library

import sqlite3

#create a sql databased connection called ship.db

connection = sqlite3.connect('ship.db')

#export the cleaned up ship dataframe as a csv called ship\_prepared with index dropped

ship.to\_csv('ship\_prepared.csv', index = False)

#export the cleaned up ship dataframe as into ship.db with index dropped

ship.to\_sql('ship.db', con = connection, index = False)

2a) We can conduct a Poisson Regression by importing the PoissonRegressor module from sklearn.linear\_model. The model has a fit function which fits the data into the PoissonRegressor module. It has parameters of X,Y and sample weight. X will be the features used for prediction and Y will be the data that you want to predict. To get the estimators of the features used in PoissonRegressor, we use the coef method available in the module. The coef method has no parameters but has to be done after fitting the data into the PoissonRegressor. The predict function is used to get the predicted value of the target variable. In this case, the predicted value will be the incidents. The predict function has a parameter X which is the features that is used to train the model.

2b)

#import linear\_model module from sklearn library

from sklearn import linear\_model

#set variable clf to be linear\_model.PoissonRegressor

clf = linear\_model.PoissonRegressor()

#print the parameters of the Poisson Regressor

print(clf.get\_params())

#fit variables X,Y into the Poisson Regressor model

clf.fit(X, Y)

#create a variable to be the estimators of the features in X

coef = clf.coef\_

#create a new dataframe called estimators which contains the results of clf.coef\_

estimator = pd.DataFrame(coef).T

#rename column names of dataframe estimator with those of X

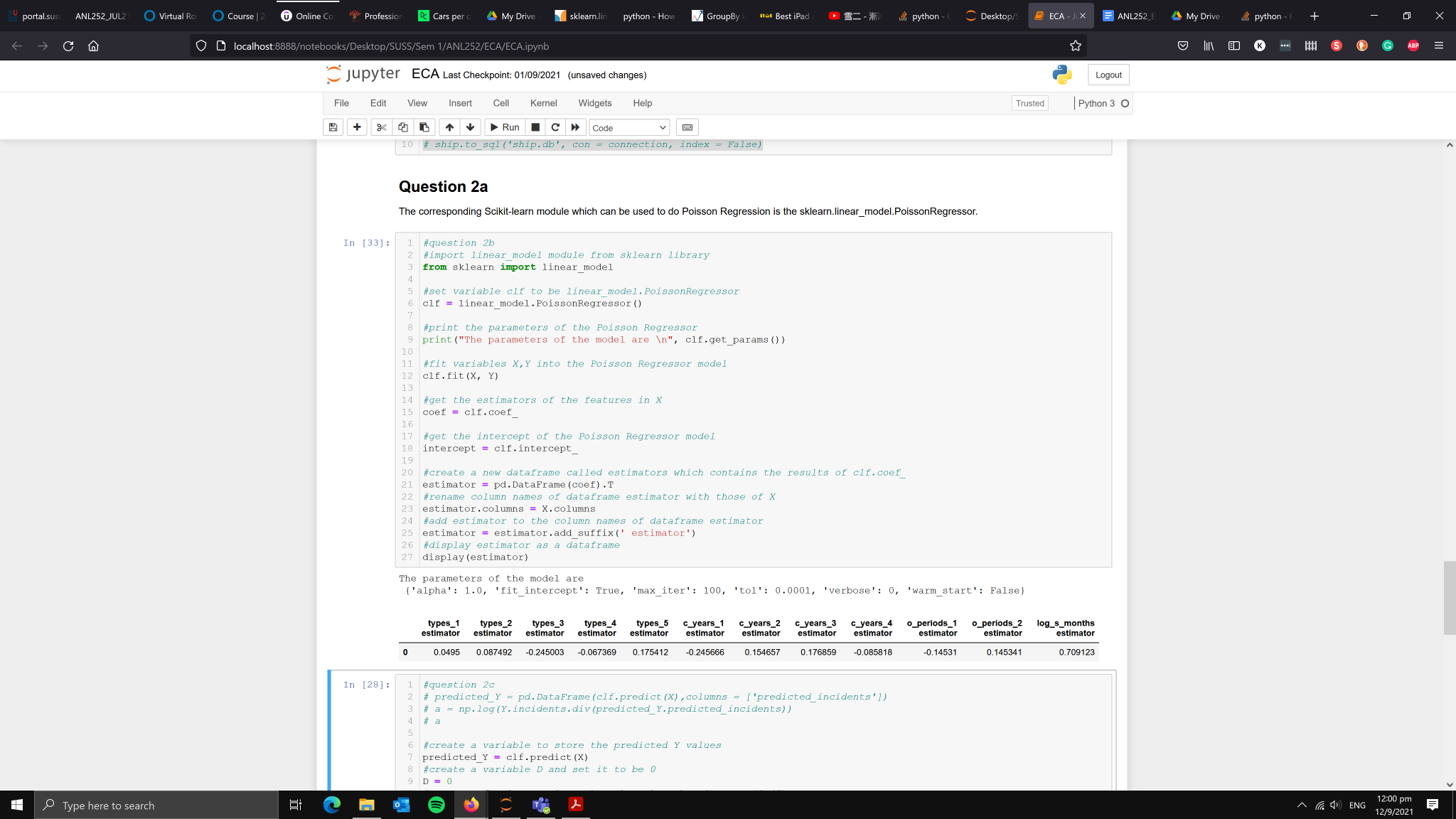
estimator.columns = X.columns

#add estimator to the column names of dataframe estimator

estimator = estimator.add\_suffix(' estimator')

#display estimator as a dataframe

display(estimator)



2c)

#create a variable to store the predicted Y values

predicted\_Y = clf.predict(X)

#create a variable D and set it to be 0

D = 0

#create a for loop to loop through each number in range(0,40)

for item in range(0,40):

#if Y = 0, enter this loop to calculate D

if float(Y.loc[item]) == 0:

D = D + predicted\_Y[item]

#if Y != 0, enter this loop to calculate D

else:

D = D + (

(Y.loc[item]\*np.log(Y.loc[item]/predicted\_Y[item]))-(Y.loc[item] - predicted\_Y[item])

)

#set D to be twice of D

D = 2\*D

#print value of D using f-string literals

print(f'D is {D:.4f}')

