



BÀI TẬP THỰC HÀNH

BỘ MÔN HỆ THỐNG THÔNG TIN



LẬP TRÌNH PHÂN TÍCH DỮ LIỆU

OCTOBER 5, 2023

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH

MỤC LỤC

BÀI TẬP TUẦN 1.....	2
1) ÔN PYTHON.....	2
2) DAY 1.....	2
3) DAY 2.....	2
4) Lab 1: Lập trình hướng đối tượng.....	2
5) Các bài Lab: Lab-01-OnPython.pdf.....	10
BÀI TẬP TUẦN 2.....	11
1) Lab 1: Pandas cơ bản	11
2) Các bài Lab:	16
BÀI TẬP TUẦN 3 - SEABORN.....	17
1) Lab 1: Pandas	17
BÀI TẬP TUẦN 4 -5: LÀM SẠCH DỮ LIỆU – KHAI THÁC DỮ LIỆU	28
1) Các bài Lab:	28
2) Pandas tiếp theo	28
3) Lab 2: TuyenSinhDaiHoc.....	39
4) BÀI 4: Mini project: Xử lý dữ liệu bảng điểm bằng Pandas	55
5) Bài 5: Kết Nối Với Sql Server	57
BÀI TẬP TUẦN 6: HỒI QUY TUYẾN TÍNH.....	59
1) Kiểm tra tuyến tính	72
2) Thực hiện &nbsp; hồi quy tuyến tính nhiều	74
3) Trong ví dụ sau, chúng tôi sẽ thực hiện hồi quy tuyến tính cho một nền kinh tế hư cấu, trong đó index_price là biến phụ thuộc và 2 biến độc lập/đầu vào là:	76

BÀI TẬP THỰC HÀNH PHÂN TÍCH DỮ LIỆU 1

BÀI TẬP TUẦN 1

- 1) ÔN PYTHON
- 2) DAY 1
- 3) DAY 2
- 4) Lab 1: Lập trình hướng đối tượng

Lab Employees

```
class Employee:
    def __init__(self,code,name,age,salary):
        self.code = code
        self.name = name
        self.age = age
        self.salary = salary
    def income(self):
        result = 0.9*12*self.salary
        return result
    def increaseSalary(self,amount):
        if amount > 0:
            self.salary = self.salary+ amount
    def display(self):
        print(f'code: {self.code}, name: {self.name}, age: {self.age}, salary: {self.salary}, income: {self.income()}\n')
```

File 2: LabEmployee.py

```
'''
Khai báo đối tượng Employee có:
Các thuộc tính (fields, states) sau
- mã số (code)
- tên (name)
- tuổi (age)
- lương (salary)
Các phương thức (behaviors, methods)
- Tổng thu nhập hằng năm sau thuế (income) biết rằng tax = 10%
- In ra thông tin của nhân viên (display)
- Tăng lương cho nhân viên (increaseSalary), biết rằng số lương tăng phải lớn hơn 0
- (Sinh viên tự viết) Giảm lương cho nhân viên (decreaseSalary), biết rằng số lương giảm phải lớn hơn 0 và không vượt quá 20% lương hiện tại
===== YÊU CẦU CHƯƠNG TRÌNH=====
Khai báo biến danh sách (list) nhân viên (dsNhanVien) để lưu trữ các nhân viên và
viết chương trình menu thực hiện các chức năng bên dưới
'''
```

```

- Opt-1: Tải danh sách nhân viên từ file dbemp_input.db
- Opt-2: Thêm nhân viên vào danh sách
- Opt-3: Hiển thị danh sách nhân viên
- Opt-4: Hiển thị thông tin của một nhân viên khi biết mã nhân viên
- Opt-5: Chỉnh sửa thông tin một nhân viên
- Opt-6: Xóa một nhân viên ra khỏi danh sách
- Opt-7: Tăng lương cho một nhân viên
- Opt-8: Giảm lương cho một nhân viên
- Opt-9: Tính số lượng nhân viên (countEmp) và xuất ra màn hình
- Opt-10: Tính tổng tiền lương của công ty phải trả hàng tháng (sumSalary) và
xuất ra màn hình
- Opt-11: Tính trung bình lương của nhân viên (avgSalary) và xuất ra màn hình
- Opt-12: Tính độ tuổi trung bình của nhân viên (avgAge) và xuất ra màn hình
- Opt-13: Tính tuổi lớn nhất của các nhân viên (maxAge) và hiển thị danh sách
nhân viên có tuổi lớn nhất
- Opt-14: Sắp xếp danh sách nhân viên tăng dần theo lương
- Opt-15: Vẽ biểu đồ tương quan lương theo độ tuổi
- Opt-16: Vẽ biểu đồ so sánh lương trung bình của các nhóm tuổi: nhỏ hơn 35, từ
35 đến 50, hơn 50 trở lên
- Opt-17: Vẽ biểu đồ thể hiện phần trăm tổng lương trên các nhóm tuổi như Opt-16
- Opt-18: Vẽ biểu đồ thể hiện phần trăm số lượng nhân viên theo các nhóm tuổi như
Opt-16
- Opt-19: Lưu danh sách nhân viên xuống file dbemp_output.db, biết rằng mỗi nhân
viên là một dòng và các thông tin nhân viên được phân cách bởi dấu '-'
- Opt-Khác: Thoát chương trình
'''

import matplotlib.pyplot as plt
import Employee as emp

menu_options = {
    1:'Load data from file',
    2:'Add new employee',
    3:'Display list of employee',
    4:'Show employee details',
    5:'Update employee information',
    6:'Delete employee',
    7:'Increase salary of employee',
    8:'Decrease salary of employee',
    9:'Show total employee a month',
    10:'Show total salary a month',
    11:'Show average of salary a month',
    12:'Show average of age',
    13:'Show maximum age',
    14:'Sort list of employee according to salary by ascending',

```

```

15: 'Draw salary according to age',
16: 'Draw average of salary chart by age group',
17: 'Draw percentage of salary by age group',
18: 'Draw percentage of total employee by age group',
19: 'Store data to file',
'Others': 'Exit program'
}

def print_menu():
    for key in menu_options.keys():
        print (key, '--', menu_options[key] )

# Khai báo biến lưu trữ những nhân viên
dsNhanVien = []

while(True):
    print_menu()
    userChoice = ''
    try:
        userChoice = int(input('Input choice: '))
    except:
        print('Invalid input, try again')
        continue
    #Check what choice was entered and act accordingly
    if userChoice == 1:
        fr = open('dbemp_input.db',mode='r',encoding='utf-8')
        for line in fr:
            stripLine = line.strip('\n')
            ds = stripLine.split(',')
            maso = ds[0]
            ten = ds[1]
            tuoi = int(ds[2])
            luong = float(ds[3])
            nv = emp.Employee(maso,ten,tuoi,luong)
            dsNhanVien.append(nv)
        fr.close()
    elif userChoice == 2:
        maso = input("Input code: ")
        ten = input("Input name: ")
        tuoi = int(input("Input age: "))
        luong = float(input("Input salary: "))
        nv = emp.Employee(maso,ten,tuoi,luong)
        dsNhanVien.append(nv)
    elif userChoice == 3:

```

```

        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            for item in dsNhanVien:
                item.display()
    elif userChoice == 4:
        #4:Show employee details
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            ma =input("Input Code:")
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
    elif userChoice == 5:
        #5:Update employee information
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            ma =input("Input Code for Update:")
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
            menu={
                1:'Update Name',
                2:'Update Age',
                3:'Update luong',
                'Others':'Thoat'
            }
            def Xuat_menu():
                for key in menu.keys():
                    print(key,'--',menu[key])
            while (True):
                Xuat_menu()
                traloi=''
                try:
                    traloi =int(input('Nhap cac tuy chon:'))
                except:
                    print('Nhap sai dinh dang, nhap lai:')
                    continue
            if traloi==1:
                ten = input("Input name: ")
                item.name =ten
                item.display()

```

```

        elif traloi ==2:
            tuoi = int(input("Input age: "))
            item.age =tuoi
            item.display()
        elif traloi ==3:
            luong = int(input("Input salary: "))
            item.salary =luong
            item.display()
        else:
            print('Ket thuc chinh sua')
            break

elif userChoice == 6:
    #6:'Delete employee'
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        ma =input("Input Code for Update:")
        for item in dsNhanVien:
            if(item.code ==ma):
                item.display()
                tl = input('Ban co chac chan xoa nhan vien nay khong
Y/N?')

                if tl =='Y':
                    #del item
                    dsNhanVien.remove(item)

        for item in dsNhanVien:
            item.display()
elif userChoice == 7:
    #7:Increase salary of employee
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        ma =input("Input Code for Update:")
        for item in dsNhanVien:
            if(item.code ==ma):
                item.display()
                luongtang = int(input('Nhap muc luong tang'))
                item.salary = item.salary + luongtang
                item.display()
elif userChoice == 8:
    #8:Decrease salary of employee
    if dsNhanVien.count==0:
        print('Danh sach rong')

```

```

        else:
            ma =input("Input Code for Update:")
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
                    luonggiam = int(input('Nhap muc luong giam'))
                    item.salary = item.salary -luonggiam
                    item.display()
    elif userChoice == 9:
        #9:'Show total employee a month'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            for item in dsNhanVien:
                tongsnv = tongsnv+1
                item.display()
            print('Tong so nhan vien =',tongsnv)
    elif userChoice == 10:
        sumSalary = 0.0
        for item in dsNhanVien:
            sumSalary = sumSalary + item.salary
        print(f'Total salary: {sumSalary}')
    elif userChoice == 11:
        #11:'Show average of salary a month'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            tongluong=0
            for item in dsNhanVien:
                tongsnv = tongsnv+1
                tongluong =tongluong +item.salary
                item.display()
            luongtb = tongluong/tongsnv
            print(f'Luong trung binh nhan vien ={luongtb}')
    elif userChoice == 12:
        #12:'Show average of age'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongtuoi=0
            tongsnv=0
            for item in dsNhanVien:

```



```

        tongsnv =tongsnv+1
        tongtuoi = tongtuoi+item.age
        item.display()
        tuoiTB = tongtuoi/tongsnv
        print(f'Tuoi trung binh nhan vien ={tuoiTB}')
elif userChoice == 13:
    #13:'Show maximum age'
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        for item in dsNhanVien:
            tuoimax=item.age
            break
        for item in dsNhanVien:
            if(item.age>tuoimax):
                tuoimax = item.age
        print('Tuoi lon nhat =',tuoimax)
elif userChoice == 14:
    #14:'Sort list of employee according to salary by ascending'
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        tongsnv=0
        for item in dsNhanVien:
            tongsnv = tongsnv+1
            item.display()
        print('Tong so nhan vien =',tongsnv)
elif userChoice == 15:
    #Draw salary according to age
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

    # Vẽ đồ thị
    plt.figure(figsize=(15,5))

    plt.title("Age and salary chart")
    plt.xlabel("Ox: age")
    plt.ylabel("Oy: salary")

    plt.plot(arrTuoi,arrLuong, "go")
    plt.show()

```

```

elif userChoice == 16:
    #16:'Draw average of salary chart by age group',
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

    # Vẽ đồ thị
    plt.figure(figsize=(15,5))

    plt.title("Age and salary chart")
    plt.xlabel("Ox: age")
    plt.ylabel("Oy: salary")

    plt.plot(arrTuoi,arrLuong, "go")
    plt.show()
elif userChoice == 17:
    #17:'Draw percentage of salary by age group'
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

    # Vẽ đồ thị
    plt.figure(figsize=(15,5))

    plt.title("Age and salary chart")
    plt.xlabel("Ox: age")
    plt.ylabel("Oy: salary")

    plt.plot(arrTuoi,arrLuong, "go")
    plt.show()
elif userChoice == 18:
    #18:'Draw percentage of total employee by age group'
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

    # Vẽ đồ thị
    plt.figure(figsize=(15,5))

```

```

plt.title("Age and salary chart")
plt.xlabel("Ox: age")
plt.ylabel("Oy: salary")

plt.plot(arrTuoi,arrLuong, "go")
plt.show()
elif userChoice == 19:
    #19:'Store data to file'
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

    # Vẽ đồ thị
    plt.figure(figsize=(15,5))

    plt.title("Age and salary chart")
    plt.xlabel("Ox: age")
    plt.ylabel("Oy: salary")

    plt.plot(arrTuoi,arrLuong, "go")
    plt.show()
else:
    print('BYE BYE')
    break

```

5) Các bài Lab: Lab-01-OnPython.pdf

BÀI TẬP TUẦN 2

1) Lab 1: Pandas cơ bản

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import random
5
6 Ô chứa mã <undefined>
7 # %% [code]
8
9 #1. Đọc File với các file trong thư mục dữ liệu. Ứng với mỗi file sinh viên thực hiện các câu sau
10 df = pd.read_csv('original_sales_data_edit.csv')
11 df
```

```
1 #3. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu đầu tiên
2 df = pd.read_csv('original_sales_data_edit.csv')
3 df.head(5)
```

```
1 #4. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu cuối cùng
2 df.tail(5)
```

```
1 #5. Chuyển kiểu dữ liệu cho 1 cột nào đó
2 df["YEAR_ID"] = df["YEAR_ID"].astype("int32")
3 df.head(5)
```

```
1 #6. Xem chiều dài của df, tương đương shape[0]
2 print('Len:', len(df))
```

```
1 #7. Xem thông tin dataframe vừa đọc được
2 df.info()
```

```
1 #8. Xem kích thước của dataframe
2 print('Shape:', df.shape)
```

```
1 #9. Hiển thị dữ liệu của cột thứ 10
2 df['PRODUCTLINE']
```

```
1 #10. Hiển thị dữ liệu của cột 1,2,3,5,6
2 df[['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'PRODUCTLINE']]
```

```
1 #11. Hiển thị 5 dòng dữ liệu đầu tiên gồm các cột 1,2,3,5,6
2 df[['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'PRODUCTLINE']].head(5)
```

```
1 #12. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số
2 df[0:5]
```

```
1 #13. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số gồm các cột 1,2,3,5,6
2 df[['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'PRODUCTLINE']][0:5]
```

```
1 #14. loại bỏ các dòng trùng nhau
2 df.drop_duplicates(inplace=True)
3 df
```

```
1 #15. Loại bỏ các dòng trống có dữ liệu của 1 cột là trống, có 2823 dòng
2 df['ADDRESSLINE2'].isna().sum() #Còn 2521 dòng
```

```
1 #16. Loại bỏ các dòng không biết của 1 cột có giá trị là Unknown, có 2521 dòng
2 df["ADDRESSLINE2"].fillna('Unknown',inplace=True)
3 df['ADDRESSLINE2'].isna().sum()
4 df
```

```
1 #17. Lấy dữ liệu của 1 cột theo dạng chuỗi
2 df["QUANTITYORDERED"]
```

```
1 #18. Lấy dữ liệu của 1 cột về một mảng
2 df["QUANTITYORDERED"].values
3
```

```
1 #20. Lấy dữ liệu từ dòng số 4 đến dòng 9
2 df[4:10]
```

```
1 #20. Lấy dữ liệu từ dòng số 4 đến dòng 9
2 df[4:10]
```

```
1 #21. Đọc dữ liệu từ dòng 4 đến dòng 9
2 df.loc[4:10]
3 df.iloc[4:10]
```

```
1 #22. Lấy thông tin tại dòng có chỉ số là 2
2 df.loc[2]
```

```
1 #23. Lấy thông tin từ dòng 4 đến dòng 10 của một số cột
2 df.loc[4:10,['QUANTITYORDERED','SALES']]
```

```
1 #24. Lấy thông tin dòng 2 đến dòng 9, từ cột 4 đến cột 7
2 df.iloc[2:9,4:7]
```

```
1 #25. Lấy dữ liệu tại chỉ số (index) là 2
2 df.iloc[2]
```

```
1 #26. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 dùng iloc
2 df.iloc[:10]
```

```
1 #27. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 gồm các cột 4 đến cột 7 dùng iloc
2 df.iloc[2:9,4:7]
```

```
1 #28. Sắp xếp dữ liệu theo sales tăng dần
2 df.sort_values(by='SALES',ascending=True)
```

```
1 #29. Sắp xếp dữ liệu theo nhiều tiêu chí
2 df.sort_values(by=['QUANTITYORDERED','PRICEEACH'],ascending=[True,False])
```

```
1 #30. Lọc dữ liệu theo 1 điều kiện
2 df[df['SALES']>5000]
3 print('Cách 2')
4 df.loc[df['SALES']>5000]
```

```
1 #31. Lọc dữ liệu theo nhiều điều kiện
2 df[(df['SALES']>5000) & (df['QUANTITYORDERED']>40)]
```

```
1 #32. Lọc giá trị và gán điều kiện dùng loc
2 df.loc[df['PRICEEACH']>=65, 'FLAG']='EXPENSIVE'
3 df.loc[df['PRICEEACH']<65, 'FLAG']='CHEAP'
4 #df['FLAG']
5 df[['PRICEEACH', 'FLAG']]#[:50]
```

```
1 #33. Viết hàm trả về giá trị có nhiều điều kiện và áp dụng hàm gán trị trả về cho 1 cột
2 def foo(x):
3     if x<10:
4         return 'BAD'
5     elif x>=10 and x<50:
6         return 'GOOD'
7     else:
8         return 'EXCELLENT'
9 df['WORTH']=df[['QUANTITYORDERED']].applymap(foo)
10 df[['QUANTITYORDERED', 'WORTH']]
```

```
1 #34. Ánh xạ giá trị tới 1 cột
2 dict_map1 = {1:'Qui_1',2:'Qui_2',3:'Qui_3',4:'Qui_4'}
3 df['QTR_ID']=df['QTR_ID'].map(dict_map1)
4 df['QTR_ID']
```

```
1 #35. Lấy những dòng dữ liệu bằng 1 điều kiện nào đó
2 df[['QUANTITYORDERED', 'PRICEEACH']].loc[df['YEAR_ID']==2003]
```

```
1 #36. Hiển thị các bản ghi có số lượng hơn 25
2 df[df['QUANTITYORDERED']>25]
3 print("Hiển thị 5 dòng")
4 Tuoitre =df[df['QUANTITYORDERED']>25]
5 Tuoitre[:5]
```

```
1 #37. Hiển thị các hoá đơn đã được giao
2 dg=df[df['STATUS'] == 'Shipped']
3 dg[:10]
```

```
1 #37. So sánh chuỗi
2 sosanh =df['STATUS'].str.contains('Shipped')
3 sosanh.head(5)
```

```
1 #38 Lấy giá trị trả về mảng
2 df['STATUS'].values
Kết quả thực thi
0KB
text/plain
array(['Shipped', 'Shipped', 'Shipped', ..., 'Resolved', 'Shipped',
      'On Hold'], dtype=object)
```

```
1 #40. Thêm 1 cột vào file dữ liệu
2 df_len =len(df)
3 ngaylap =[random.randrange(2003,2005,1) for i in range(df_len)]
4 df['ORDERDATE']=ngaylap
5 df.tail(5)
```

```
1 #41. Thêm 1 cột (DaGiao)vào dữ liệu theo tiêu chí nếu điều kiện thoả thì giá trị mặc định là True, ngược lại là False.
2 df['DaGiao'] =df['STATUS']=='Shipped'
3 df.head(5)
```

```
1 #42. Tạo 1 cột mới có giá trị rỗng
2 df['TONGTIEN']=None
3 df
```

```
1 #44. Sửa giá trị của cột
2 df['QTR_ID'] =None
3 df['QTR_ID']='Quy '
4 df.head(5)
```

```
1 #45. Xóa cột trong dataframe
2 df.drop(['TONGTIEN'],axis=1)
3 df.head(5)
```

```
1 #46. Xóa bản ghi theo chỉ số
2 df.drop([0,1]) #Xoá bản ghi có chỉ số 1 và 2
```

```
1 #47.Sử dụng hàm describe() để thống kê dữ liệu
2 df.describe()
```

```
1 #48. Xem thống kê trên từng cột
2 df['STATUS'].value_counts()
```

```
1 #49. Vẽ đồ thị xem phân bố giá trị của 1 trường trong dataframe
2 df['STATUS'].value_counts().plot(kind='bar')
```

```
1 #50. Tạo mới dataframe từ các python list
2 peoples = {'name': ['Nguyễn Văn Hiếu', 'Hiếu Nguyễn Văn'], 'age': [28, 28], 'website': ['https://blog.luyencode.net', None]}
3 df1 = pd.DataFrame(peoples)
4 print(df1)
```

```
1 #Tạo mới dataframe từ các python list
2 txts = ['chỗ này ăn cũng khá ngon', 'ngon, nhất định sẽ quay lại', 'thái độ phục vụ quá tệ']
3 labels = [1, 1, 0]
4 df1 = pd.DataFrame()
5 df1['txt'] = txts
6 df1['label'] = labels
7 print(df1)
```

```
1 #51. Sắp xếp dataframe
2 # Sắp xếp df tăng dần theo cột nào đó
3 df1 = pd.DataFrame({'name': ['Nam', 'Hiếu', 'Mai', 'Hoa'], 'age': [18,18,17,19]})
4 print('Before sort\n', df1)
5 df1 = df1.sort_values('age', ascending=True)
6 print('After sort\n', df1)
```

```

1 #52. Nối 2 dataframe
2 # Gộp 2 dataframe
3 df_1 = pd.DataFrame({'name': ['Hiếu'], 'age': [18], 'gender': ['male']})
4 df_2 = pd.DataFrame({'name': ['Nam', 'Mai', 'Hoa'], 'age': [15,17,19]})
5 df_3 = df_1.append(df_2, sort=True)
6 print(df_3)

```

```

1 #53. Xáo trộn các bản ghi trong dataframe
2 df1 = pd.DataFrame({'name': ['Hiếu', 'Nam', 'Mai', 'Hoa'], 'age': [18,15,17,19]})
3 print('Before shuffle\n', df)
4 df1 = df.sample(frac=1).reset_index(drop=True)
5 print('After shuffle\n', df1)

```

```

1 #54. Lưu dataframe về file csv
2 df1.to_csv('KHACHHANG.csv')

```

```

1 #55. Tối ưu bộ nhớ khi dùng pandas
2 import numpy as np # linear algebra
3 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
4
5 def reduce_mem_usage(df):
6     """ iterate through all the columns of a dataframe and modify the data type
7     to reduce memory usage.
8     """
9     start_mem = df.memory_usage().sum() / 1024**2
10    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))
11
12    for col in df.columns:
13        col_type = df[col].dtype
14
15        if col_type != object and col_type.name != 'category' and 'datetime' not in col_type.name:
16            c_min = df[col].min()
17            c_max = df[col].max()
18            if str(col_type)[:3] == 'int':
19                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
20                    df[col] = df[col].astype(np.int8)
21                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
22                    df[col] = df[col].astype(np.int16)
23                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
24                    df[col] = df[col].astype(np.int32)
25                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
26                    df[col] = df[col].astype(np.int64)
27            else:
28                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
29                    df[col] = df[col].astype(np.float16)
30                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
31                    df[col] = df[col].astype(np.float32)
32                else:
33                    df[col] = df[col].astype(np.float64)
34            elif 'datetime' not in col_type.name:
35                df[col] = df[col].astype('category')

```



```

34         elif 'datetime' not in col_type.name:
35             df[col] = df[col].astype('category')
36
37     end_mem = df.memory_usage().sum() / 1024**2
38     print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
39     print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))
40
41     return df
42 df

```

2) Các bài Lab:

BÀI TẬP TUẦN 3 - SEABORN

1) Lab 1: Pandas

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

```
#loading dataset
#Iris dataset contains five columns such as Petal Length, Petal Width, Sepal
Length, Sepal Width and Species Type. Iris is a flowering plant, the researchers
have measured various features of the different iris flowers and recorded them
digitally.
data =sns.load_dataset("iris")
#draw lineplot
sns.lineplot(x="sepal_length",y="sepal_width",data=data)
```

```
#Seaborn is built on the top of Matplotlib. It means that Seaborn can be used
with Matplotlib.
#Using Seaborn with Matplotlib
#We will be using the above example and will add the title to the plot using the
Matplotlib.
#Draw lineplot
sns.lineplot(x="sepal_length",y="sepal_width",data=data)
#setting the title using Matplotlib
plt.title('Title using Matplotlib Function')
plt.show()
```

```
#Setting the xlim and ylim
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# setting the x limit of the plot vẽ bd với x=5
plt.xlim(5)
plt.show()
```

```
#Customizing Seaborn Plots
#set_style() method is used to set the aesthetic of the plot. It means it affects
things like the color of the axes, whether the grid is active or not, or other
aesthetic elements. There are five themes available in Seaborn:
#darkgrid, whitegrid, dark, white, ticks
#set_style(style=None, rc=None)
```

```
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# changing the theme to dark
sns.set_style("dark")
plt.show()
```

```
#Removal of Spines
#Spines are the lines noting the data boundaries and connecting the axis tick
marks. It can be removed using the despine() method.
#sns.despine(left = True)
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Removing the spines
sns.despine(left=True)
plt.show()
```

```
#Changing the figure Size
# changing the figure size
plt.figure(figsize = (2, 4))
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Removing the spines
sns.despine()
plt.show()
```

```
#Scaling the plots
#It can be done using the set_context() method. It allows us to override default
parameters. This affects things like the size of the labels, lines, and other
elements of the plot, but not the overall style. The base context is "notebook",
and the other contexts are "paper", "talk", and "poster". font_scale sets the
font size.
#set_context(context=None, font_scale=1, rc=None)
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Setting the scale of the plot
sns.set_context("poster")
plt.show()
```

```
#Setting the Style Temporarily
#axes_style() method is used to set the style temporarily. It is used along with
the with statement.
#axes_style(style=None, rc=None)
```

```
def plot():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
with sns.axes_style('darkgrid'):
    # Adding the subplot
    plt.subplot(211)
    plot()
plt.subplot(212)
plot()
```

```
#Color Palette
#color_palette() method is used to give colors to the plot.
#palplot() is used to deal with the color palettes and plots the color palette as
a horizontal array.
# current colot palette
palette = sns.color_palette()

# plots the color palette as a
# horizontal array
sns.palplot(palette)

plt.show()
```

```
#Diverging Color Palette: uses two different colors where each color depicts
different points ranging from a common point in either direction. Consider a
range of -10 to 10 so the value from -10 to 0 takes one color and values from 0
to 10 take another.
# current colot palette
palette = sns.color_palette('PiYG', 11)

# diverging color palette
sns.palplot(palette)

plt.show()
```

```
#Sequential Color Palette: is used where the distribution ranges from a lower
value to a higher value. To do this add the character 's' to the color passed in
the color palette.
# current colot palette
palette = sns.color_palette('Greens', 11)

# sequential color palette
sns.palplot(palette)
```

```
plt.show()
```

```
#Setting the default Color Palette
#set_palette() method is used to set the default color palette for all the plots.
The arguments for both color_palette() and set_palette() is same. set_palette()
changes the default matplotlib parameters.
def plot():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)

# setting the default color palette
sns.set_palette('vlag')
plt.subplot(211)

# plotting with the color palette
# as vlag
plot()

# setting another default color palette
sns.set_palette('Accent')
plt.subplot(212)
plot()

plt.show()
```

```
#Multiple plots with Seaborn
#Using Matplotlib: add_axes(), subplot(), and subplot2grid().
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Creating a new figure with width = 5 inches
# and height = 4 inches
fig = plt.figure(figsize =(5, 4))

# Creating first axes for the figure
ax1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])

# plotting the graph
graph()

# Creating second axes for the figure
ax2 = fig.add_axes([0.5, 0.5, 0.3, 0.3])

# plotting the graph
```

```
graph()

plt.show()
```

```
#Using subplot() method
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)

# Adding the subplot at the specified
# grid position
plt.subplot(121)
graph()

# Adding the subplot at the specified
# grid position
plt.subplot(122)
graph()

plt.show()
```

```
#Using subplot() method
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)

# Adding the subplot at the specified
# grid position
plt.subplot(121)
graph()

# Adding the subplot at the specified
# grid position
plt.subplot(122)
graph()

plt.show()
```

```
#Using subplot2grid() method
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)

# adding the subplots
axes1 = plt.subplot2grid (
```

```
(7, 1), (0, 0), rowspan = 2, colspan = 1)
graph()
axes2 = plt.subplot2grid (
    (7, 1), (2, 0), rowspan = 2, colspan = 1)
graph()

axes3 = plt.subplot2grid (
    (7, 1), (4, 0), rowspan = 2, colspan = 1)
graph()

#Using Seaborn: Using FacetGrid() method
#FacetGrid class helps in visualizing distribution of one variable as well as the
relationship between multiple variables separately within subsets of your dataset
using multiple panels.
#seaborn.FacetGrid( data, \*\*kwargs)
plot = sns.FacetGrid(data, col="species")
plot.map(plt.plot, "sepal_width")

plt.show()
```

```
#Using PairGrid() method
#Subplot grid for plotting pairwise relationships in a dataset.
#This class maps each variable in a dataset onto a column and row in a grid of
multiple axes. Different axes-level plotting functions can be used to draw
bivariate plots in the upper and lower triangles, and the marginal distribution
of each variable can be shown on the diagonal.
#seaborn.PairGrid( data, \*\*kwargs)
data = sns.load_dataset("flights")

plot = sns.PairGrid(data)
plot.map(plt.plot)

plt.show()
```

```
#Creating Different Types of Plots
#Relplot()
#This function provides us the access to some other different axes-level
functions which shows the relationships between two variables with semantic
mappings of subsets. It is plotted using the relplot() method
#seaborn.relplot(x=None, y=None, data=None, **kwargs)
# loading dataset
data = sns.load_dataset("iris")
```

```
# creating the relplot
sns.relplot(x='sepal_width', y='species', data=data)

plt.show()
```

```
#The scatter plot is a mainstay of statistical visualization. It depicts the
joint distribution of two variables using a cloud of points, where each point
represents an observation in the dataset. This depiction allows the eye to infer
a substantial amount of information about whether there is any meaningful
relationship between them. It is plotted using the scatterplot() method
#seaborn.scatterplot(x=None, y=None, data=None, **kwargs)
sns.scatterplot(x='sepal_length', y='sepal_width', data=data)
plt.show()
```

```
#The scatter plot is a mainstay of statistical visualization. It depicts the
joint distribution of two variables using a cloud of points, where each point
represents an observation in the dataset. This depiction allows the eye to infer
a substantial amount of information about whether there is any meaningful
relationship between them. It is plotted using the scatterplot() method
#seaborn.scatterplot(x=None, y=None, data=None, **kwargs)
sns.scatterplot(x='sepal_length', y='sepal_width', data=data)
plt.show()
```

```
#Line Plot: For certain datasets, you may want to consider changes as a function
of time in one variable, or as a similarly continuous variable. In this case,
drawing a line-plot is a better option. It is plotted using the lineplot()
method.
#seaborn.lineplot(x=None, y=None, data=None, **kwargs)
sns.lineplot(x='sepal_length', y='species', data=data)
plt.show()
```

```
#Categorical Plots are used where we have to visualize relationship between two
numerical values. A more specialized approach can be used if one of the main
variable is categorical which means such variables that take on a fixed and
limited number of possible values.
#A barplot is basically used to aggregate the categorical data according to some
methods and by default its the mean. It can also be understood as a visualization
of the group by action. To use this plot we choose a categorical column for the x
axis and a numerical column for the y axis and we see that it creates a plot
taking a mean per categorical column. It can be created using the barplot()
method.
```



```
#barplot([x, y, hue, data, order, hue_order, ...])
sns.barplot(x='species', y='sepal_length', data=data)
plt.show()
```

#A countplot basically counts the categories and returns a count of their occurrences. It is one of the most simple plots provided by the seaborn library. It can be created using the countplot() method.

```
#countplot([x, y, hue, data, order, ...])
sns.countplot(x='species', data=data)
plt.show()
```

#A boxplot is sometimes known as the box and whisker plot. It shows the distribution of the quantitative data that represents the comparisons between variables. boxplot shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution i.e. the dots indicating the presence of outliers. It is created using the boxplot() method.

```
#boxplot([x, y, hue, data, order, hue_order, ...])
sns.boxplot(x='species', y='sepal_width', data=data)
plt.show()
```

#Violinplot

#It is similar to the boxplot except that it provides a higher, more advanced visualization and uses the kernel density estimation to give a better description about the data distribution. It is created using the violinplot() method.

```
#violinplot([x, y, hue, data, order, ...])
sns.violinplot(x='species', y='sepal_width', data=data)
plt.show()
```

#Stripplot: It basically creates a scatter plot based on the category. It is created using the stripplot() method.

```
#stripplot([x, y, hue, data, order, ...])
sns.stripplot(x='species', y='sepal_width', data=data)
plt.show()
```

#Swarmplot is very similar to the stripplot except the fact that the points are adjusted so that they do not overlap. Some people also like combining the idea of a violin plot and a stripplot to form this plot. One drawback to using swarmplot is that sometimes they don't scale well to really large numbers and takes a lot of computation to arrange them. So in case we want to visualize a swarmplot properly we can plot it on top of a violinplot. It is plotted using the swarmplot() method.

```
#swarmplot([x, y, hue, data, order, ...])
sns.swarmplot(x='species', y='sepal_width', data=data)
plt.show()
```

```
#Factorplot is the most general of all these plots and provides a parameter
called kind to choose the kind of plot we want thus saving us from the trouble of
writing these plots separately. The kind parameter can be bar, violin, swarm etc.
It is plotted using the factorplot() method.
#sns.factorplot([x, y, hue, data, row, col, ...])
# loading dataset
data = sns.load_dataset("iris")
sns.factorplot(x='species', y='sepal_width', data=data)
plt.show()
#A histogram is basically used to represent data provided in a form of some
groups. It is accurate method for the graphical representation of numerical data
distribution. It can be plotted using the histplot() function.
#histplot(data=None, *, x=None, y=None, hue=None, **kwargs)
sns.histplot(x='species', y='sepal_width', data=data)
plt.show()
```

```
#Distplot is used basically for univariant set of observations and visualizes it
through a histogram i.e. only one observation and hence we choose one particular
column of the dataset. It is potted using the distplot() method.
#distplot(a[, bins, hist, kde, rug, fit, ...])
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = sns.load_dataset("iris")
sns.distplot(data['sepal_width'])
plt.show()
```

```
#Jointplot is used to draw a plot of two variables with bivariate and univariate
graphs. It basically combines two different plots. It is plotted using the
jointplot() method.
#jointplot(x, y[, data, kind, stat_func, ...])
sns.jointplot(x='species', y='sepal_width', data=data)
plt.show()
```

```
#Pairplot represents pairwise relation across the entire dataframe and supports
an additional argument called hue for categorical separation. What it does
basically is create a jointplot between every possible numerical column and takes
a while if the dataframe is really huge. It is plotted using the pairplot()
method.
#pairplot(data[, hue, hue_order, palette, ...])
sns.pairplot(data=data, hue='species')
```

```
plt.show()
```

```
#Rugplot plots datapoints in an array as sticks on an axis. Just like a distplot it takes a single column. Instead of drawing a histogram it creates dashes all across the plot. If you compare it with the joinplot you can see that what a jointplot does is that it counts the dashes and shows it as bins. It is plotted using the rugplot() method.
```

```
#rugplot(a[, height, axis, ax])
sns.rugplot(data=data)
plt.show()
```

```
#KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.
```

```
#seaborn.kdeplot(x=None, *, y=None, vertical=False, palette=None, **kwargs)
sns.kdeplot(x='sepal_length', y='sepal_width', data=data)
plt.show()
```

```
#The regression plots are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses. Regression plots as the name suggests creates a regression line between two parameters and helps to visualize their linear relationships.
```

```
#lmpplot() method can be understood as a function that basically creates a linear model plot. It creates a scatter plot with a linear fit on top of it.
```

```
#seaborn.lmpplot(x, y, data, hue=None, col=None, row=None, **kwargs)
```

```
# loading dataset
```

```
data = sns.load_dataset("tips")
sns.lmpplot(x='total_bill', y='tip', data=data)
plt.show()
```

```
#regplot() method is also similar to lmpplot which creates linear regression model.
```

```
#seaborn.regplot( x, y, data=None, x_estimator=None, **kwargs)
```

```
# loading dataset
```

```
data = sns.load_dataset("tips")

sns.regplot(x='total_bill', y='tip', data=data)
```

```
plt.show()
```

```
#Heatmap is defined as a graphical representation of data using colors to
visualize the value of the matrix. In this, to represent more common values or
higher activities brighter colors basically reddish colors are used and to
represent less common or activity values, darker colors are preferred. it can be
plotted using the heatmap() function.
#seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None,
annot_kws=None, linewidths=0, linecolor='white', cbar=True, **kwargs)
# loading dataset
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("tips")

# correlation between the different parameters
tc = data.corr()

sns.heatmap(tc)
plt.show()
```

```
#The clustermap() function of seaborn plots the hierarchically-clustered heatmap
of the given matrix dataset. Clustering simply means grouping data based on
relationship among the variables in the data.
#clustermap(data, *, pivot_kws=None, **kwargs)
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = sns.load_dataset("tips")

# correlation between the different parameters
tc = data.corr()

sns.clustermap(tc)
plt.show()
```

```
--loi
```

BÀI TẬP TUẦN 4 -5: LÀM SẠCH DỮ LIỆU – KHAI THÁC DỮ LIỆU

- 1) Các bài Lab:
 - a) Lab4-data-wragling.pdf
 - b) Lab4_LamSachDuLieu_new.pdf
- 2) Pandas tiếp theo

```
#=====QUÁ TRÌNH EXPLANATORY DATA ANALYSIS=====
#Đây là quá trình khai thác thông tin tri thức từ dữ liệu thông qua biểu đồ.
#Khi vẽ biểu đồ chúng ta cần đặt rá các câu hỏi
#1. Có bao nhiêu biến tham gia vào biểu đồ
#2. Loại (định tính, định lượng) của từng biến số
#3. Biểu đồ biểu diễn bao nhiêu thông tin
#4. Tri thức gì được rút trích ra từ biểu đồ
#5. Tri thức được rút trích có liên quan gì đến nghiệp vụ
#=====
#Bước 1: Xử lý dữ liệu cơ bản theo yêu cầu
#=====
#1.1. Đọc dữ liệu
#1.2. Loại bỏ dòng dữ liệu trống
#1.3. Loại bỏ dòng dữ liệu bị trùng
#1.4. Kiểm tra các dữ liệu thiếu bằng chart
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
#1.1. Đọc dữ liệu
#df = pd.read_csv('original_sales_data_edit.csv')
df =pd.read_csv('original_sales_data_edit.csv',encoding='utf-8',
header=0,delimiter=',')
df
#Kết quả: 2824 dòng
```

```
1 #1.2. Loại bỏ dòng dữ liệu rỗng
2 df.dropna(how='all',inplace=True)
3 df #2824 dòng
```

```
#1.3. Loại bỏ dữ liệu trùng, biết rằng dữ liệu trùng là dữ liệu có đồng thời
ORDERNUMBER và OrDERDATE như nhau
#Kiểm tra lại nghiệp vụ này
df.drop_duplicates(inplace=True)
df #2824 dòng
```

```
#1.3. Loại bỏ dữ liệu trùng, biết rằng dữ liệu trùng là dữ liệu có đồng thời
ORDERNUMBER và OrDERDATE như nhau
#Kiểm tra lại nghiệp vụ này
df.drop_duplicates(inplace=True)
df #2824 dòng
```

```
#1.4. Kiểm tra dữ liệu thiếu bằng chart
# Cách 2: Trực quan dữ liệu thiếu với Seaborn Displot
plt.figure(figsize=(10,6))
sns.displot(data=df.isna().melt(value_name="missing"),
y="variable",
hue="missing",
multiple="fill",
aspect=1.25)
plt.savefig("my_missing_value_2.png",dpi=100)
```

```
#1.4.1. Điền thiếu dữ liệu với dữ liệu định tính
#1.4.1.1. Với dữ liệu biểu diễn dạng chuỗi thì thay bằng Unknown
#1.4.1.2. Với dữ liệu biểu diễn dạng số thì thay bằng 0
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#1.1. Đọc dữ liệu
df =pd.read_csv('original_sales_data_edit.csv', encoding='UTF8',header=0,
delimiter=',')
```

```
#1.2. Loại bỏ dòng dữ liệu rỗng
print("Dữ liệu chưa loại bỏ")
df.info
df
#Dữ liệu sau khi loại bỏ
print("Dữ liệu sau khi loại bỏ")
df.dropna(how='all',inplace=True)
df
```

LOAI BO DU LIEU TRUNG BIET RANG DU LIEU TRUNG LA DU LIEU CO DONG THOI ORDERNUMBER
V ORDERDATE NHU NHAU // KIEM TRA LAI NGHIEP VU NAY

```
#1.3. Loại bỏ dữ liệu trùng
#2823 dòng
df.drop_duplicates(inplace=True)
df
```

```
df.drop_duplicates(subset=['ORDERNUMBER','ORDERDATE'],inplace=True)
df[['ORDERNUMBER','ORDERDATE']]#Kết quả là 2733
```

```
#1.4. Kiểm tra các dữ liệu thiếu bằng chart
#Trực quan dữ liệu thiếu với Seaborn HeatMap, cột màu đậm đang bị thiếu dữ liệu
plt.figure(figsize=(10,6))
sns.heatmap(df.isna().transpose(),cmap="YlGnBu",cbar_kws={'label':'Missing
Data'})
plt.savefig("my_missing_value_1.png",dpi=100)
```

```
#Cách 2: Trực quan dữ liệu thiếu bằng Seaborn Displot
plt.figure(figsize=(10,6))
sns.displot(data=df.isna().melt(value_name="missing"),y="variable",
hue="missing",multiple="fill",aspect=1.25)
plt.savefig("my_missing_value_2.png",dpi=100)
```

```
#1.4.1. Điền thiếu dữ liệu với dữ liệu định tính
#1.4.1.1. Với dữ liệu biểu diễn dạng chuỗi thì thay dữ liệu thiếu bằng Unknown
#1.4.1.2. Với dữ liệu biểu diễn dạng số thì thay bằng 0
df['ADDRESSLINE2'].fillna('Unknown', inplace=True)
df['STATE'].fillna('Unknow',inplace=True)
df['TERRITORY'].fillna('Unknown',inplace=True)
df['POSTALCODE'].fillna(0,inplace=True)
print("Xem lại thông tin")
df[['ADDRESSLINE2','STATE','TERRITORY','POSTALCODE']]
```

```
#1.5. Tách 1 cột thành 2 cột. Sau đó xoá cột bị tách
df[['PAYMENTLASTNAME','PAYMENTFIRSTNAME']]=df['PAYMENTFULLNAME'].str.split('
',expand=True)
df=df.drop('PAYMENTFULLNAME',axis=1)
df[['PAYMENTLASTNAME','PAYMENTFIRSTNAME']]
```

```
#1.6. Lưu dữ liệu thành file mới
df.to_csv('processed_sales_data.csv',sep=',',encoding='utf-8',index=False)
```

```
#1.1.1. Cho biết tổng số lượng sản phẩm bán được của mỗi năm
sns.barplot(x='YEAR_ID',
            y='QUANTITYORDERED',data=df,errorbar=None,estimator=sum)
plt.show()
```

```
#1.1.2.Có bao nhiêu đơn hàng theo Dealsize
labels=df['DEALSIZE'].value_counts().index
```

```
values=df['DEALSIZE'].value_counts().values
colors=sns.color_palette('pastel')
plt.pie(values,labels=labels, colors=colors, autopct='%1.1f%%', shadow=True)
plt.show()
```

1.1.3.HOAc SU DUNG NANG CAO VOI NHIEU TUY CHINH TRONG HAM TONG HOP

#1.1.3. Cho biết tỉ lệ giá trị SALES theo DEALSIZE

```
gb=df.groupby(['DEALSIZE'])['ORDERNUMBER'].agg(['count'])
data=list(gb['count'])
labels=gb.index
colors=sns.color_palette('pastel')
plt.pie(values,labels=labels, colors=colors, autopct='%1.1f%%', shadow=True)
plt.show()
```

CHO BIET GIA TRI SALES THEO NGAY

#1.1.7. Cho biết tổng giá trị SALES theo ngày

```
sns.lineplot(x="ORDERDATE",y="SALES",data=df, estimator=sum)
```

#1.1.8. Cho biết giá trị SALES trung bình theo tháng năm

```
#default estimator=mean
```

```
sns.lineplot(x="MONTH_ID", y="SALES", hue='YEAR_ID',data=df)
plt.show()
```

#1.1.8. Cho biết giá trị SALES trung bình theo tháng năm

```
#default estimator=mean
```

```
sns.lineplot(x="MONTH_ID", y="SALES", hue='YEAR_ID',data=df)
plt.show()
```

#1.1.8.a Cho biết giá trị SALES theo tháng năm

```
from numpy import count_nonzero
```

```
sns.lineplot (
x="MONTH_ID",y="ORDERNUMBER",hue='YEAR_ID',data=df,estimator=count_nonzero)
plt.show()
```

#1.1.9. Cho biết số lượng hoá đơn theo tháng, năm

```
#Dùng biểu đồ Line
```

```
from numpy import count_nonzero
```

```
sns.lineplot(x='MONTH_ID',y='ORDERNUMBER',hue='YEAR_ID',data=df,estimator=count_n
onzero)
plt.show()
```



```
#Cách 2: Dùng barplot
sns.barplot(x='STATUS', y='ORDERNUMBER', hue='YEAR_ID', data=df,
            errorbar=None, estimator=count_nonzero)
plt.show()
```

CHO BIẾT GIA TRI DON HANG THEO TRANG THAI (STATUS) THEO NHOM CAC NAM(YEAR_ID)

```
#Dùng biểu đồ barplot
sns.barplot(x='STATUS', y='SALES', hue='YEAR_ID', data=df, errorbar=None)
plt.show()
```

TONG GIA TRI DON HANG THEO TRANG THAI(STATUS) THEO NHOM CAC NAM (YEAR_ID)

#1.2.1. Cho biết tổng giá trị đơn hàng theo từng trạng thái (STATUS) của mỗi năm

```
sns.barplot(x='STATUS', y='SALES', hue='YEAR_ID', data=df, errorbar=None, estimator=sum)
plt.show()
```

SO LUONG HOA DON GIUA CAC DEALSIZE THEO YEAR_ID

#1.2.3. Cho biết số lượng hoá đơn giữa các nhóm DEALSIZE theo từng năm YEAR_ID

```
gb=df.groupby(['DEALSIZE', 'YEAR_ID'])['ORDERNUMBER'].count().unstack()
gb.plot(kind='bar', stacked=True)
#just add a title and rotate the x-axis labels to the horizontal
plt.title('Number of Orders in DEALSIZE, YEAR_ID')
plt.xticks(rotation=0, ha='center')
plt.show()
```

TRUNG BINH SALES CAC NHO STATUS THEO DEALSIZE

#1.2.4. Cho biết trung bình SALES theo từng STATUS và DEALSIZE

```
gb=df.groupby(['STATUS', 'DEALSIZE'])['SALES'].mean().unstack()
gb.plot(kind='barh', stacked=True)
#just add a title and rotate the x-axis labels to the horizontal
plt.title(' TRUNG BINH SALES CAC NHOM THEO DEALSIZE')
plt.xticks(rotation=0, ha='center')
plt.show()
```

3. MÔ TẢ DỮ LIỆU

#3.1. Mô tả dữ liệu cột QuantityOrdered

```
df['QUANTITYORDERED'].describe()
```

MO TA DU LIEU CUA QUANTITYORDERED, PRICEEACH, SALES

```
#3.2. Mô tả dữ liệu của QUANTITYORDERED, PRICEEACH
df[['QUANTITYORDERED', 'PRICEEACH', 'SALES']].describe()
```

```
MO TA DU LIEU SALES THEO NHOM DEALSIZE
#Mô tả số lượng bán hàng theo từng DEALSIZE
df.groupby('DEALSIZE')['QUANTITYORDERED'].describe()
```

```
PHẦN 4: KHAI THÁC SỰ PHÂN PHỐI DỮ LIỆU, CHỈ DÙNG TRÊN BIẾN ĐỊNH LƯỢNG
#4.1. VE BIEU DO HISTOGRAM CUA QUANTITYORDERED
df['QUANTITYORDERED'].hist()
plt.show()
```

```
#HOAC NANG CAO HON VOI NHIEU TUY CHON
# sns displot(df,x="QUANTITYORDERED",kind="kde")
# plt.show()
```

```
#HOAC NANG CAO HON VOI NHIEU TUY CHON
sns.displot(df,x="QUANTITYORDERED",kind="kde")
plt.show()
```

```
CHO BIET XAC SUAT XAY RA TRÊN MỖI ĐOẠN
#4.2. VE BIEU DO HIS CUA QUANTITYORDERED THEO DEALSIZE //PHAN PHOI HIS CUA
QUANTITYORDERED THEO NHOM DEALSIZE
#màu xanh lá cây ổn định nhất
sns.displot(df, x="QUANTITYORDERED",hue="DEALSIZE",kind="kde")
plt.show()
```

```
#4.3. VE BIEU DO HIS CUA QUANTITYORDERED THEO DEALSIZE //PHAN PHOI HIS CUA
QUANTITYORDERED THEO NHOM DEALSIZE
sns.displot(df, x="QUANTITYORDERED",hue="DEALSIZE",kind="ecdf")
plt.show()
```

```
#4.4. VE BIEU DO HIS CUA QUANTITYORDERED THEO DEALSIZE //PHAN PHOI HIS CUA
QUANTITYORDERED THEO NHOM DEALSIZE
sns.displot(df, x="QUANTITYORDERED",hue="DEALSIZE",kind="hist")
plt.show()
```

```
VE BIEU DO JOINPLOT CUA QUANTITYORDERED VA PRICEEACH
```

```
#4.5. VE BIEU DO JOINPLOT CUA QUANTITYORDERED VA PRICEEACH
```

```
sns.jointplot(data=df,x='PRICEEACH',y='QUANTITYORDERED',kind='kde',color='y')
```

```
#4.6. VẼ BIỂU ĐỒ PAIRPLOT CỦA QUANTITYORDERED,PRICEEACH,SALES
sns.pairplot(df[['QUANTITYORDERED','PRICEEACH','SALES']],diag_kind='hist',kind='kde')
plt.show()
```

BIỂU DIỄN TƯƠNG QUAN BIẾN SỐ THEO TỪNG CẤP

```
#4.8. Vẽ trực quan số lượng sản phẩm phân phối theo năm của QuantityOrdered (catplot)
sns.catplot(x='YEAR_ID',y='QUANTITYORDERED',data=df)
plt.show()
# Hỏi câu hỏi:
```

VE TRUC QUAN SO LUONG SAN PHAM PHAN PHOI THEO NAM CUA QUANTITYORDERED

```
#4.9. Vẽ trực quan số lượng sản phẩm phân phối theo năm của cột QUANTITYORDERED
sns.boxplot(data=df['QUANTITYORDERED'])
plt.show()
```

BA CÁCH BIỂU DIỄN

```
#4.10.1.BIỂU THỊ CHO NGOẠI BIÊN
#BIỂU DIỄN (BOXPLOT,VIOLIN) CỦA QUANTITYORDERED TRÊN CUNG CHART
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title('Simple plot')
#fig, axes = plt.subplots(2, 2, subplot_kw=dict(projection="polar"))
fig,axes =plt.subplots(nrows=1,ncols=2,sharey=True,figsize(6,4))
sns.boxplot(data=df['QUANTITYORDERED'],ax=axes[0])
sns.violinplot(data=df['QUANTITYORDERED'],ax=axes[1])
plt.show()
-----loi
```

BIỂU DIỄN BOXPLOT CỦA QUANTITYORDERED THEO NHÓM DEALSIZE

```
sns.boxplot(x='DEALSIZE',y='QUANTITYORDERED',data=df)
plt.show()
```

BIỂU DIỄN CATPLOT CỦA QUANTITYORDERED THEO NHÓM DEALSIZE

```
sns.catplot(x='YEAR_ID',y='QUANTITYORDERED',hue='DEALSIZE',data=df)
plt.show()
```

BIỂU DIỄN BOXPLOT CỦA QUANTITYORDERED THEO NĂM, NHÓM DEALSIZE

```
sns.boxplot(x='YEAR_ID',y='QUANTITYORDERED',hue='DEALSIZE',data=df)
plt.show()
```

BIEU DIEN BOXPLOT CUA QUANTITYORDERED,PRICEEACH

```
sns.boxplot(data=df[['QUANTITYORDERED','PRICEEACH']])
plt.show()
```

DO XIEN CUA PHAN PHOI (SKEW) CUA QUANTITYORDERED,PRICEEACH

```
df[['QUANTITYORDERED','PRICEEACH']].skew()
```

DO NHON CUA PHAN PHOI (KURTOSIS) CUA QUANTITYORDERED , PRICEEACH

```
df[['QUANTITYORDERED','PRICEEACH']].kurtosis()
```

KIEM TRA TINH CHUAN (NORMAL DISTRIBUTION) CUA QUANTITYORDERED,PRICE,SALES

```
from scipy import stats
stats.probplot(df['QUANTITYORDERED'],plot=sns.mpl.pyplot)
plt.show()
```

THUC HIEN NORAMLIZE DU LIEU QUANTITYORDERED

```
from sklearn import preprocessing
min_max_scaler=preprocessing.MinMaxScaler()
df[['QUANTITYORDERED']]= min_max_scaler.fit_transform(df[['QUANTITYORDERED']])
sns.displot(df, x="QUANTITYORDERED", kind="kde")
plt.show()
```

THUC HIEN STANDARDIZATION DU LIEU QUANTITYORDERED

```
from sklearn.preprocessing import StandardScaler
scaler =StandardScaler()
df[['QUANTITYORDERED']]=scaler.fit_transform(df[['QUANTITYORDERED']])
sns.displot(df,x="QUANTITYORDERED",kind="kde")
plt.show()
```

```
#c2
from scipy import stats
df['QUANTITYORDERED']=stats.zscore(df['QUANTITYORDERED'])
stats.zscore(df['QUANTITYORDERED'])
sns.displot(df,x="QUANTITYORDERED",kind="kde")
```

```
plt.show()
```

```
MA TRAN TUONG QUAN TUYEN TINH(PERSON) CUA CAC CAC QUANTITYORDERED,PRICEEACH,SALES
df[['QUANTITYORDERED','PRICEEACH','SALES']].corr()
#hoat df[['QUANTITYORDERED','PRICEEACH','SALES']].cov()
```

VE BIEU DO HEATMAP TUONG QUAN GIUA CAC CAP 'QUANTITYORDERED','PRICEEACH','SALES'

```
sns.heatmap(df[['QUANTITYORDERED','PRICEEACH','SALES']].corr(),vmax=1.0,square=False).xaxis.tick_top()
```

TUONG QUAN CUA BIEN QUANTITYORDERED,PRICEEACH,SALES THEO NHOM DEALSIZE

```
df.groupby('DEALSIZE')[['QUANTITYORDERED','PRICEEACH','SALES']].corr()
```

VE BIEU DO CHO BIAET GIA TRI MIN ,MAX CUA SO LUONG SAN PHAM TRONG MOI DON HANG THEO QUY,NAM

```
gb=df.groupby(['QTR_ID','YEAR_ID'])['QUANTITYORDERED'].agg(['min','max'])
gb.plot(kind='bar',stacked=False)
plt.show()
```

VE BIEU DO CHO BIET TONG SO LUONG SAN PHAM THEO THANG

```
sns.lineplot(x="MONTH_ID",y="QUANTITYORDERED",hue='YEAR_ID',data=df,estimator=sum)
plt.show()
```

VE BIEU DO CHO BIET QUAN HE GIUA SALES (Oy) VA DON GIA(Ox)// DUNG BD SCATTER

```
sns.lmplot(data=df,x='PRICEEACH',y='SALES',fit_reg=False)
plt.show()
```

VE BIEU DO CHO BIET QUAN HE GIUA SALES (Oy) VA DON GIA(Ox) THEO DEALSIZE

```
sns.lmplot(data=df,x='PRICEEACH',y='SALES',hue='DEALSIZE',fit_reg=False)
plt.show()
```

VE BIEU DO CHO BIET QUAN HE GIUA SALES (Oy) VA DON GIA(Ox) THEO DEALSIZE QUA CAC NAM (YEAR_ID) THEO COT VA THEO QUY QUA DONG

```
sns.lmplot(data=df,x='PRICEEACH',y='SALES',hue='DEALSIZE',col='YEAR_ID',row='QTR_ID',fit_reg=False)
plt.show()
```

```
#sap xep du lieu theo sales tang dan
df.sort_values(by='SALES',ascending=True)
```

SAP XEP DU LIEU TANG DAN NHUNG KHI TRUNG LAP SE XET THEO MUC GIAM DAN O COT KHAC

```
df.sort_values(by=['QUANTITYORDERED','PRICEEACH'],ascending=[True,False])
```

LỌC DATAFARME

```
#Hiển thị các dòng có Sales>5000
df[df['SALES']>5000]
```

#Cách 2:

```
df.loc[df['SALES']>5000]
```

ĐIỀU KIỆN GỘP

```
#Hiển thị các dòng có Sales>5000 và QuantityOrdered>40
df[(df['SALES']>5000) & (df['QUANTITYORDERED']>40)]
```

TẠO CỘT MỚI VÀ GÁN THUỘC TÍNH MỚI TRÊN BẢNG

```
#Lọc ra các dòng có Priceeach<65, nếu có thay giá trị bằng Cheap, ngược lại thay bằng Expensive
df.loc[df['PRICEEACH']<65,'FLAG']='CHEAP'
df.loc[df['PRICEEACH']>=65,'FLAG']='EXPENSIVE'
df[['PRICEEACH','FLAG']]
```

```
#Viết hàm foo(x) nếu x<10 trả về Bad, nếu x>=10 và <50 trả về Good, ngược lại trả về Excellent
```

```
def foo(x):
    if x < 10 :
        return 'BAD'
    elif x >= 10 and x < 50:
        return 'GODD'
    else:
        return 'EXCELLENT'
```

```
#Áp dụng cho cột month
```

```
df['MONTH']= df[['QUANTITYORDERED']].applymap(foo)
```

```
#Hiển thị kết quả
```

```
df[['QUANTITYORDERED','MONTH']]
```

SO SÁNH CỘT

```
#Viết hàm so sánh giá trị nếu x<=y trả về giá trị YES, ngược lại là NO
```

```
def ftrust(x,y):
    if x<=y:
```

```
        return 'YES'
    else:
        return 'NO'
#Áp dụng hàm để gán giá trị trả về cho TRUST
df['TRUST']=list(map(ftrust, df['PRICEEACH'],df['MSRP']))
```

```
#Hiển thị kết quả sau khi gọi hàm
df[['PRICEEACH','MSRP','TRUST']]
```

```
#Thay đổi giá trị cho cột QTR_ID là Q1 nếu giá trị là 1, Q2 nếu giá trị là 2,...
dict_map= {1:'Q1', 2:'Q2', 3:'Q3', 4:'Q4'}
df['QTR_ID']= df['QTR_ID'].map(dict_map)
```

XÁC ĐỊNH CÁC HÀM TỔNG HỢP TRÊN CÁC BIẾN ĐỊNH LƯỢNG VÀ ĐỊNH TÍNH

```
#Hiển thị giá trị tổng , giá trị nhỏ nhất cho cột QUANTITYORDERED, giá trị nhỏ nhất và lớn nhất và trung bình cho cột PRICEEACH, giá trị nhỏ nhất và trung bình cho cột SALES
df.aggregate({'QUANTITYORDERED':['sum','min'],'PRICEEACH':['min','max','mean'],'SALES':['min','mean']})
```

GROUP BY: NHÓM

```
#Hiển thị giá trị tổng , giá trị nhỏ nhất cho cột QUANTITYORDERED, giá trị nhỏ nhất và lớn nhất và trung bình cho cột PRICEEACH, giá trị nhỏ nhất và trung bình cho cột SALES
df.aggregate({'QUANTITYORDERED':['sum','min'],'PRICEEACH':['min','max','mean'],'SALES':['min','mean']})
```

```
#Thống kê tổng, trung vị, độ lệch chuẩn của SALES theo từng nhóm DEALSIZE
df.groupby(['DEALSIZE'])['SALES'].agg(['sum','mean','std'])
```

```
#Thống kê tổng, trung vị, độ lệch chuẩn của SALES theo từng nhóm DEALSIZE, cùng DEALSIZE thì nhóm theo QTR_ID
df.groupby(['DEALSIZE','QTR_ID'])['SALES'].agg(['sum','mean','std'])
```

XUẤT BẢNG TÍNH THEO THUỘC TÍNH CỦA BẢNG

```
pd.pivot_table(df,values='SALES',columns='YEAR_ID',aggfunc=['sum','mean'])
```

```
#Thống kê theo dạng bảng Pivot 2 chiều: Thống kê tổng và trung bình số lượng QUANTITYORDERED và SALES theo Year_ID
pd.pivot_table(df,values=['SALES','QUANTITYORDERED'],columns='YEAR_ID',aggfunc=['sum','mean'])
```

3) Lab 2: TuyenSinhDaiHoc

```
'''
Phần 1: thao tác cơ bản
Bước 1: Xử lý cơ bản
1. Xác định số lượng yếu tố (biến số) tham gia vào yêu cầu
2. Thu thập dữ liệu (data collection)
3. Tổng quan dữ liệu VD: df.info()...
4. Xử lý cơ bản:
    - Loại bỏ dòng rỗng
    - Loại bỏ dòng trùng
    - Khảo sát dữ liệu thiếu
    - Xử lý dữ liệu thiếu
5. Kiểm tra lại dữ liệu
'''
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#Đọc file
df = pd.read_csv('dulieuxettuyendaihoc.csv',header=0,delimiter=',',
encoding='utf-8')
#Đọc 5 dòng dữ liệu đầu tiên
df.head(5)
```

```
#Xem thông tin tổng quan
df.info()
# Tổng quan dữ liệu
```

Biến	Kiểu dữ liệu	Loại	Thang đo	
Malop	Chuỗi ký tự	Định tính		

```
#Lấy thông tin các cột
df = df[['T5', 'T6', 'GT', 'DT', 'KV', 'KT', 'NGONNGU', 'TOANLOGICPHANTICH',
'GIAIQUYETVANDE', 'NGAYTHI', 'DINH HUONGNGHENGHIEP']]
```

```
# Đổi tên cột
df.rename(columns={'TOANLOGICPHANTICH': 'LOGIC', 'GIAIQUYETVANDE': 'UNGXU',
'DINH HUONGNGHENGHIEP': 'HUONGNGHIEP'}, inplace=True)
```



```
df.head(5)
```

```
#Xóa bỏ các dòng dữ liệu rỗng
df.dropna(how='all',inplace=True)
```

```
# dùng heatmap để trực quan dữ liệu bị thiếu
plt.figure(figsize=(10,6))
sns.heatmap(df.isna().transpose(),cmap='YlGnBu',
cbar_kws={'label':'Dữ liệu thiếu'})
plt.savefig('missingdata.png',dpi=100)
'''
```

Note: Với dữ liệu bị thiếu:

1. Cần xác định biến số nào bị thiếu
2. Mức độ thiếu dữ liệu
3. Có cần phải xử lý không

```
'''
```

```
# Điền giá trị thiếu
df['DT'].fillna('KINH',inplace=True)
# Lưu ý: Với biến định tính ta có thể thay bằng giá trị yếu vị (mode)
# df['DT'].fillna(df['DT'].mode()[0],inplace=True)
```

```
# Điền thiếu giá trị phần NGONNGU bằng 0 (nếu có)
df['NGONNGU'].fillna(0,inplace=True)
# Điền thiếu giá trị phần LOGIC bằng trung bình (nếu có)
df['LOGIC'].fillna(df['LOGIC'].mean(), inplace=True)
# Điền thiếu giá trị phần UNGXU bằng trung vị (nếu có)
df['UNGXU'].fillna(df['UNGXU'].median(),inplace=True)
# Lưu ý: Với biến định lượng thì ta nên thay bằng trung vị
```

```
df.head(5)
```

```
'''
Phần 2: Kỹ thuật Feature Engineering (thường dùng cho Machine Learning)
Nếu chỉ là xử lý phân tích dữ liệu thì ta gọi là New Attribute

Đây là kỹ thuật tạo thêm hoặc biến đổi số liệu sẵn có thành các biến số mới phù
hợp với nghiệp vụ
để phân tích

Tạo biến TBTOAN: trung bình toán lớp 12
'''
```

```
df['TBTUAN'] = (df['T5'] + df['T6']) / 2
df.head(5)
```

```
# Tạo biến XEPLAI: đánh giá môn toán dựa trên df['TBTUAN']
df.loc[df['TBTUAN'] < 5.0, 'XEPLAI'] = 'FAIL'

df.loc[(df['TBTUAN'] >= 5.0) & (df['TBTUAN'] < 7.0), 'XEPLAI'] = 'FAIR'

df.loc[(df['TBTUAN'] >= 7.0) & (df['TBTUAN'] < 9.0), 'XEPLAI'] = 'GOOD'

df.loc[(df['TBTUAN'] >= 9.0), 'XEPLAI'] = 'EXCEL'

df.head(5)
```

```
#Xem thông tin 5 dòng đầu gồm các cột TBTUAN, XEPLAI
df[['TBTUAN', 'XEPLAI']].head(5)
```

```
'''
Tạo biến nhóm khối thi NHOMKT thỏa mãn
A1: G1
C: G3
D1: G3
A: G1
B: G2
'''

dict_map = {'A1': 'G1', 'C': 'G3', 'D1': 'G3', 'A': 'G1', 'B': 'G2'}
df['NHOMKT'] = df['KT'].map(dict_map)
```

```
df[['KT', 'NHOMKT']].head(5)
```

```
'''BỮA SAU ngày 25/9
Tạo biến số điểm cộng: CONG

Nếu khối thi thuộc nhóm G1, G2 và TBTUAN >= 5.0 thì là 1.0
Ngược lại thì là 0.0
'''

def fplus(x, y):
    if (x == 'G1' or x == 'G2') and (y >= 5.0):
        return 1.0
    else:
        return 0.0

df['CONG'] = list(map(fplus, df['NHOMKT'], df['TBTUAN']))
```

```
df[['TBTOAN','NHOMKT','CONG']].head(5)
```

```
#Phần 3: Trực quan hóa dữ liệu
'''
Để trực quan số liệu ta cần lưu ý: Mục đích, sự phối hợp giữa các loại biến để
chọn lựa biểu đồ phù hợp đi kèm số liệu
trực quan
Định tính: bar, pie
Định lượng: line , histogram, box-plot, scatter
'''
'''
Hãy trực quan số lượng học sinh theo giới tính
'''
# Biểu đồ bar
sns.countplot(x='GT', data=df)
plt.show()
```

```
# Lưu ý
# Các biến dùng để phân nhóm, gom nhóm thông thường là biến định tính và nằm ở
các thang đo mức 1,2,3,4
# tương ứng định danh, phân loại, thứ bậc và khoảng
```

```
# Dựa trên biểu đồ DT cho biết tạo sao ta không phân tích theo nhóm DT : vì đa số
là dân tộc kinh
# Tương tự cho các cột KV, DT, KT
#DT
sns.countplot(x='DT', data=df)
plt.show()
```

```
sns.countplot(x='KV', data=df)
plt.show()
```

```
sns.countplot(x='KT', data=df)
plt.show()
```

```
'''
Hãy so sánh số lượng học sinh đăng ký khối thi dựa trên nhóm giới tính
'''
sns.countplot(x='KT',hue='GT',data=df)
plt.show()
```

```
'''
```

```
Làm tương tự cho các nhóm biến định tính (KV,KT)
Hãy cho biết khối A có sinh viên khu vực nào đăng ký cao nhất
Trả lời: KV1
'''
sns.countplot(x='KV',hue='KT',data=df)
plt.show()
```

```
'''
Hãy so sánh điểm trung bình NGONNGU theo nhóm giới tính
'''
sns.barplot(x='GT',y='NGONNGU',data=df,errorbar=None)
plt.show()
```

```
# Hãy so sánh điểm LOGIC theo nhóm KT (nhóm khối thi)
sns.barplot(x='NHOMKT',y='LOGIC',data=df,errorbar=None)
plt.show()
```

```
'''
So sánh điểm trung bình của NGONNGU theo nhóm GT dựa trên KT
'''
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None)
plt.show()
```

```
# So sánh sai số trên NGONNGU theo nhóm GT theo KT
# Sai số càng cao độ tin cậy càng thấp
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df)
plt.show()
```

```
'''
So sánh điểm cao nhất của NGONNGU theo nhóm GT theo KT
Lưu ý: không để estimator thì mặc định là mean
estimator: count, min max sum std, mean (default)
'''
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=max)
plt.show()
```

```
'''
So sánh điểm cao nhất của NGONNGU theo nhóm GT theo KT
Lưu ý: không để estimator thì mặc định là mean
estimator: count, min max sum std, mean (default)
'''
```

```
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=max)
plt.show()
```

```
'''
So sánh điểm cao nhất của NGONNGU theo nhóm GT theo KT
Lưu ý: không để estimator thì mặc định là mean

estimator: count, min max sum std, mean (default)
'''

sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=max)
plt.show()
```

```
'''
Khi biến định tính dùng làm nhóm tổng hợp có nhiều hơn 2 giá trị thì ta cần dùng
hàm tổng hợp thông qua thư viện numpy
'''

sns.barplot(x='KV',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=np.max)
plt.show()
```

```
'''
Lưu ý:
- Với biến định tính thì ta chỉ có 1 hàm tổng hợp là hàm COUNT, MODE
- Với định lượng thì ta có thể sử dụng các hàm tổng hợp như: COUNT, MAX, MIN,
MEAN, MEDIAN, MODE, SUM, STD
'''
```

```
'''
Biểu đồ PIE
Mục đích: Trực quan dữ liệu theo nhóm tỉ lệ phần trăm
'''

gb = df.groupby(['KT'])['KT'].agg(['count'])
# group by trên nhóm khối thi trên biến khối thi và dùng hàm count
```

```
'''
Biểu đồ PIE
Mục đích: Trực quan dữ liệu theo nhóm tỉ lệ phần trăm
'''

gb = df.groupby(['KT'])['KT'].agg(['count'])
# group by trên nhóm khối thi trên biến khối thi và dùng hàm count

labels = gb.index
data = list(gb['count'])
```

```
colors = sns.color_palette('pastel') # tạo bảng màu

plt.pie(data, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True)
plt.show()
```

```
'''
Trực quan tỉ lệ % tổng điểm CONG trên từng nhóm khu vực
# coi khu vực nào dc cộng điểm nhiều nhất
'''

gb = df.groupby(['KV'])['CONG'].agg(['sum'])

labels = gb.index
data = list(gb['sum'])

colors = sns.color_palette('pastel') # tạo bảng màu

plt.pie(data, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True)
plt.show()
```

```
'''
KHí trực quan dữ liệu ta cần lưu ý đến loại biến đang tham gia vào trực quan

Thông thường việc chọn lựa biểu đồ sẽ căn cứ dựa trên ý nghĩa nghiệp vụ và sự
phối hợp giữa các loại biến như:
- Định tính kết hợp định tính
- Định tính kết hợp định lượng
- Định lượng kết hợp định lượng
'''
```

```
'''
Biểu đồ line thường dùng để tổng hợp dữ liệu theo trục "Thời gian" hoặc "có thứ
tự"
Ví dụ: tổng hợp trung bình điểm cộng theo các năm thi
'''

sns.lineplot(x='NGAYTHI', y='CONG', data=df)
plt.show()
```

```
# Trực quan dữ liệu tổng điểm CONG dựa theo năm bằng biểu đồ line
```

```
sns.lineplot(x='NGAYTHI',y='CONG',data=df,estimator=sum)
plt.show()
```

```
# tổng hợp tổng điểm cộng theo các năm thi trên từng nhóm giới tính bằng biểu đồ
line
sns.lineplot(x='NGAYTHI',y='CONG',hue='GT',data=df,estimator=sum)
plt.show()
```

```
'''
Buổi 3
Bước 1: mô tả biến định lượng
'''
df['NGONNGU'].describe()
# Giải thích ý nghĩa các đại lượng
# Độ lệch chuẩn (std) bằng căn bậc 2 giá trị phương sai, độ lệch chuẩn và phương
sai thể hiện mức độ biến thiên, biến động
# sự đa dạng của tập dữ liệu số. Độ lệch chuẩn càng cao thì tập dữ liệu biến động
mạnh => mức độ đa dạng nhiều và ngược lại thì tập dữ liệu sẽ ổn định hơn

# Tứ phân vị
# Q1 : 25% -> Có 25% dữ liệu nhỏ hơn giá trị Q1
# Q2: 50% (median trung vị) -> giá trị này cho biết có 50% sv nhỏ hơn Q2 và lớn
hơn Q2
# Q3: 75% -> có 25% số lượng lớn hơn Q3
# Q1 - Q3 là khoảng IQR: khoảng mà các dữ liệu được diễn ra dc coi là thông
thường (50%)
```

```
df[['NGONNGU','LOGIC','UNGXU']].describe()
```

```
'''
CV = std/mean (Coefficient of variant)
So sánh mức độ ổn định của điểm số
'''
cvNN = df['NGONNGU'].std()/df['NGONNGU'].mean()
cvLogic = df['LOGIC'].std()/df['LOGIC'].mean()
cvUngXu = df['UNGXU'].std()/df['UNGXU'].mean()
print('cvNN: ', cvNN)
print('cvLogic: ', cvLogic)
print('cvUngXu: ', cvUngXu)
# df[['NGONNGU','LOGIC','UNGXU']].std()/df[['NGONNGU','LOGIC','UNGXU']].mean()
```

```
df.groupby('GT')[['NGONNGU','LOGIC','UNGXU']].describe()
```

```
'''
Histogram cho biết xác suất xảy ra của biến cố trong khoảng giá trị dữ liệu nào
nhiều nhất
'''
```

```
df['NGONNGU'].hist(bins=20)
plt.show()
```

```
# Hướng dẫn khi vẽ bins trong histogram
# Lưu ý: khi số lượng bins khác nhau sẽ dẫn đến hình dạng của histogram khác nhau
df['NGONNGU'].hist(bins=14)
plt.show()
```

```
'''
Nâng cao hơn histogram thì ra khám phá dạng phân phối xác suất
Làm mịn với phân phối xác suất
'''
sns.displot(df,x='NGONNGU',kind='kde')
plt.show()
```

```
sns.displot(data= df[['NGONNGU','LOGIC','UNGXU']],kind='kde')
plt.show()
# Hãy cho biết phân phối của biến số nào gần với phân phối chuẩn hơn => logic với
ứng xử
```

```
sns.displot(df,x='NGONNGU',hue='GT',kind='kde')
plt.show()
# Câu hỏi: nhóm giới tính nào gần hơn: F gần hơn
```

```
'''
Đây là biểu đồ quan trọng trong việc phân tích dữ liệu định lượng
Biểu đồ này cung cấp các thông tin quan trọng như
1. Q1: Tứ phân vị 25%
2. Q2: Tứ phân vị 50% (median)
3. Q3: Tứ phân vị 75%
4. Độ lớn IQR = |Q3-Q1|
5. Lower bound: Q1 - 1.5*IQR
6. Upper bound = Q3 + 1.5*IQR
7. Các ngoại biên, bất thường (outlier) cần xử lý trong dữ liệu
Outlier: là điểm dữ liệu khác biệt quá nhiều so với đa số
Hướng dẫn
+ Tính khoảng nghi ngờ chứa outliers
'''
```



```

+ Tính khoảng chắc chắn chứa outliers
'''
sns.boxplot(data=df['LOGIC'],orient="h")
print('lower bound = ', df['LOGIC'].quantile(0.25) -
1.5*(df['LOGIC'].quantile(0.75) - df['LOGIC'].quantile(0.25)))
print('upper bound = ', df['LOGIC'].quantile(0.75) +
1.5*(df['LOGIC'].quantile(0.75) - df['LOGIC'].quantile(0.25)))
IQR = df['LOGIC'].quantile(0.75) - df['LOGIC'].quantile(0.25)
print('IQR',IQR)
print('ngoai bien dưới', 1.625 -1.5*IQR)
print('ngoai bien trên', 6.625 +1.5*IQR)

```

```

# Tính khoảng giá trị nghi ngờ bất thường
# Tính khoảng giá trị được cho là bất thường
# Tính xem có bao nhiêu sinh viên có điểm thi là bất thường

```

```

sns.boxplot(data=df[['NGONNGU','LOGIC','UNGXU']],orient='h')

```

```

# Hãy cho biết điểm số môn nào không xảy ra bất thường => NGONNGU

```

```

sns.boxplot(x='NGONNGU',y='KT',data=df,orient='h')
plt.show()
# Câu hỏi: khối thi nào có lower bound trùng với phân vị thứ 1 (Q1) => khối B

```

```

sns.boxplot(x='NGONNGU',y='KV',data=df,orient='h')
plt.show()

```

```

sns.boxplot(x='KT',y='NGONNGU',hue='GT',data=df)

```

```

sns.boxplot(x='KT',y='NGONNGU',hue='GT',data=df)
plt.show()
# câu hỏi: xác định các biểu đồ bất thường
# Khối A1 không đủ dữ liệu để vẽ

```

```

'''
Skewness = độ xiên, độ lớn (trị tuyệt đối) cho biết mức độ dữ liệu lệch nhiều hay
ít so với đường công phân phối chuẩn
Cho biết xác suất được phân bố lệch về phía nào nhiều
Trị tuyệt đối giá trị càng lớn thì dữ liệu phân phối nghiêng càng nhiều (lệch)

Diễn giải cho skewness
skewnes > 0 tức là mean > median: ta gọi là positive skewness

```

hay lệch phải, tức là giá trị ngoại biên outliers nhận giá trị lớn sẽ đẩy giá trị trung bình về phía cuối

skewnes < 0 tức là mean < median: ta gọi là negative skewness hay lệch trái, tức là giá trị outliers nhận giá trị nhỏ sẽ đẩy giá trị trung bình về phía đầu

skewness = 0 tức là mean = median = mode: phân phối không lệch còn được gọi là phân phối đối xứng

'''

```
df['NGONNGU'].skew()
```

'''

Note: Khi ptich dữ liệu với các phương pháp có liên quan phân phối chuẩn thì cần kiểm tra skewness

nếu dl quá lệch so với phân phối chuẩn thì ta cần điều chỉnh bằng các hàm transform cho bớt lệch

đặc biệt là phân tích hồi quy

'''

```
df[['NGONNGU', 'LOGIC', 'UNGXU']].skew()
```

'''

Kurtosis: Độ nhọn, trị tuyệt đối cho biết mức độ nhọn của phân phối

Độ lớn của kurtosis càng gần 3 thì fit

Dưới 3 thì fat

Trên 3 thì thin

Thông thường để đánh giá hình dáng độ nhọn ta dùng đại lượng excess kurtosis (theo Pearson) - 3

+ Nếu excess > 0 -> thin

+ Nếu excess = 0 -> fit

+ Nếu excess < 0 -> fat

Trong pandas sử dụng Fisher's Kurtosis tức là đã chuẩn hóa giá trị về excess kurtosis theo mean = 0

+ Trị tuyệt đối excess kurtosis càng cao thì mức độ thin, fat càng lớn

Lưu ý : với phân phối chuẩn thì excess kurtosis = 0, skewness = 0

'''

```
df[['NGONNGU']].kurtosis()
```

Câu hỏi: biến ngôn ngữ có độ nhọn như thế nào

Trả lời là: fat

```
df[['NGONNGU','LOGIC','UNGXU']].kurtosis()
```

```
sns.displot(data=df[['LOGIC','UNGXU']],kind='kde')
plt.show()
# Nhìn biểu đồ cho biết ý nghĩa kurtosis cho LOGIC UNGXU

'''
Kiểm định phân phối chuẩn
'''

from scipy import stats
stats.probplot(df['NGONNGU'],plot=sns.mpl.pyplot)
plt.show()
# không có phân phối chuẩn
```

```
'''
Phân tích sự tác động (ảnh hưởng) qua lại giữa 2 biến định lượng
'''
'''
Buổi 3:
Hiệp phương sai: co-variance
Giá trị co-variance > 0 thì 2 biến có tương quan thuận (đồng biến)
Giá trị co-variance < 0 thì 2 biến có tương quan nghịch (nghịch biến)
Độ lớn (trị tuyệt đối của giá trị) càng lớn thì mức độ quan hệ (tương quan) càng
chặt chẽ

Ma trận hiệp phương sai: co-variance matrix
'''
df[['T5','T6']].cov()
```

```
# So sánh mức độ tương quan giữa T5 so với LOGIC và T6 so với LOGIC
df[['T5','T6','LOGIC']].cov()
```

```
'''
Với phương pháp so sánh tương quan bằng co-variance thì ta không đo lường được
cường độ
tương quan giữa 2 biến định lượng.

Pearson Correlation: tương quan tuyến tính
r nằm trong khoảng [-1,1]
r = 0 => Không tương quan
r < 0: Tương quan nghịch
```

```
r > 0: tương quan thuận
|r| (độ lớn) càng gần 1 thì tương quan càng cao
|r| < 0.5 thì tương quan thấp
    [0.5,0.65]: Khá
    [0.65,0.75]: Tốt
    [0.75,0.9]: Rất tốt
    > 0.9: hoàn hảo
```

Ma trận tương quan: correlation matrix

Lưu ý: được sử dụng khảo sát tương quan tuyến tính nhằm phân tích mối quan hệ tuyến tính giữa 2 biến định lượng

```
'''
df[['T5','T6']].corr()
```

```
'''
Trực quan hóa tương quan tuyến tính giữa 2 biến định lượng
Khám phá tương quan tuyến tính của 2 biến định lượng
thông qua biểu đồ phân tán (scatter)
'''
sns.lmplot(data=df, x='T5',y='T6', fit_reg=True)
plt.show()
```

```
# Sinh viên tự khám phá độ tương quan giữa biến T6 và UNGXU
df[['T6','UNGXU']].corr()
sns.lmplot(data=df, x='T6',y='UNGXU', fit_reg=True)
plt.show()
```

```
df[['T6','UNGXU']].corr()
```

```
df[['T6','UNGXU','NGONNGU','LOGIC','UNGXU']].corr()
```

```
sns.heatmap(df[['T5','T6','UNGXU','NGONNGU','LOGIC','UNGXU']].corr(),vmax=1.0,square=False).xaxis.tick_top()
```

```
# Biểu đồ tổng hợp khám phá tổng hợp nhiều biến định lượng
sns.pairplot(df[['T5','T6','NGONNGU','LOGIC','UNGXU']],
diag_kind='kde',kind='reg')
plt.show()
```

```
# Trực quan tương quan tuyến tính theo nhóm (định tính) giữa 2 biến định lượng
sns.lmplot(data=df,x='T5',y='T6',hue='GT',fit_reg=True)
plt.show()
```

BÀI 3

2) Sinh viên làm các câu hỏi sau theo dữ liệu của mỗi sinh viên

1. Đọc File với các file trong thư mục dữ liệu. Ứng với mỗi file sinh viên thực hiện các câu sau. Hiển thị toàn bộ dữ liệu của file dữ liệu đã đọc
2. Đọc file với chỉ định không có header, dòng header của chúng ta đã biến thành 1 bản ghi dữ liệu. Hiển thị toàn bộ dữ liệu của file dữ liệu đã đọc
3. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu đầu tiên
4. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu cuối cùng
5. Chuyển kiểu dữ liệu cho 1 cột nào đó
6. Xem chiều dài của df, tương đương `shape[0]`
`print('Len:', len(df))`
7. Xem thông tin dataframe vừa đọc được
`df.info()`
8. Xem kích thước của dataframe
`print('Shape:', df.shape)`
9. Hiển thị dữ liệu của cột thứ 3
`df['tên cột']`
10. Hiển thị dữ liệu của cột 1,2,3,5,6
`df[['Tên cột 1', 'Tên cột 2', 'Tên cột 3']]`
11. Hiển thị 5 dòng dữ liệu đầu tiên gồm các cột 1,2,3,5,6
`df[['Tên cột 1', 'Tên cột 2', 'Tên cột 3']].head(5)`
12. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số
`df[0:5]`
13. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số gồm các cột 1,2,3,5,6
`df[['Tên cột 1', 'Tên cột 2', 'Tên cột 3']][0:5]`
14. Loại bỏ các dòng trùng nhau
`df.drop_duplicates(inplace=True)`
15. Loại bỏ các dòng trống có dữ liệu của 1 cột là trống
`df['ADDRESSLINE2'].isna().sum()`
16. Loại bỏ các dòng của 1 cột có giá trị là không biết (“unknown”)
`df["ADDRESSLINE2"].fillna('Unknown',inplace=True)`
`df['ADDRESSLINE2'].isna().sum()`
17. Lấy dữ liệu của 1 cột theo dạng chuỗi
`df["QUANTITYORDERED"]`
18. Lấy dữ liệu của 1 cột về một mảng
`df["QUANTITYORDERED"].values`
19. Lấy dữ liệu của 1 cột về một mảng

- `df["QUANTITYORDERED"].values`
20. Lấy dữ liệu từ dòng số 4 đến dòng 9
`df[4:10]`
 21. Đọc dữ liệu từ dòng 4 đến dòng 9
`df.loc[4:10]`
`df.iloc[4:10]`
 22. Lấy thông tin tại dòng có chỉ số là 2
`df.loc[2]`
 23. Lấy thông tin từ dòng 4 đến dòng 10 của một số cột
`df.loc[4:10,['QUANTITYORDERED','SALES']]`
 24. Lấy thông tin dòng 2 đến dòng 9, từ cột 4 đến cột 7
`df.iloc[2:9,4:7]`
 25. Lấy dữ liệu tại chỉ số (index) là 2
`df.iloc[2]`
 26. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 dùng `iloc`
`df.iloc[:10]`
 27. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 gồm các cột 4 đến cột 7 dùng `iloc`
`df.iloc[2:9,4:7]`
 28. Sắp xếp dữ liệu theo 1 cột tăng dần
`df.sort_values(by='SALES',ascending=True)`
 29. Sắp xếp dữ liệu theo nhiều tiêu chí
`df.sort_values(by=['QUANTITYORDERED','PRICEEACH'],ascending=[True,False])`
 30. Lọc dữ liệu theo 1 điều kiện
`df[df['SALES']>5000]`
`df.loc[df['SALES']>5000]`
 31. Lọc dữ liệu theo nhiều điều kiện
`df[(df['SALES']>5000) & (df['QUANTITYORDERED']>40)]`
 32. Lọc giá trị và gán điều kiện dùng `loc`
`df.loc[df['PRICEEACH']>=65,'FLAG']='EXPENSIVE'`
`df.loc[df['PRICEEACH']<65,'FLAG']='CHEAP'`
`#df['FLAG']`
`df[['PRICEEACH','FLAG']]#[:50]`
 33. Viết hàm trả về giá trị có nhiều điều kiện và áp dụng hàm gán trị trả về cho 1 cột
`def foo(x):`
`if x<10:`

```

        return 'BAD'
    elif x>=10 and x<50:
        return 'GOOD'
    else:
        return 'EXCELLENT'
df['WORTH']=df[['QUANTITYORDERED']].applymap(foo)

```

```
df[['QUANTITYORDERED','WORTH']]
```

34. Ánh xạ giá trị tới 1 cột

```

dict_map = {1:'Qui_1',2:'Qui_2',3:'Qui_3',4:'Qui_4'}
df['QTR_ID']=df['QTR_ID'].map(dict_map)
df['QTR_ID']

```

35. Lấy những dòng dữ liệu bằng 1 điều kiện nào đó

```
df[['QUANTITYORDERED','PRICEEACH']].loc[df['YEAR_ID']==2003]
```

36. Hiện thị các bản ghi có cột kiểu số hơn 25

```

df[df['age']<25]
Tuoitre =df[df['age']<25]
Tuoitre[:5]

```

37. Sinh viên tự nghĩ ra các câu hỏi đọc dữ liệu theo các điều kiện và thực hiện lại các câu hỏi đó. Yêu cầu sinh viên ghi lại và lệnh thực hiện các câu hỏi đó.

38. Thực hiện 1 ví dụ để lấy giá trị của một cột trả về dưới dạng numpy array trong thư viện pandas python.

39. Sử dụng thư viện random để sinh ngẫu nhiên một list năm sinh và thêm vào dataframe.

40. Thêm 1 cột vào file dữ liệu

41. Thêm 1 cột vào dữ liệu theo tiêu chí nếu điều kiện thỏa thì giá trị mặc định là True, ngược lại là False.

42. Tạo 1 cột mới có giá trị rỗng

43. Thêm 1 bản ghi trong dataframe

44. Sửa giá trị của cột

45. Xóa cột trong dataframe

46. Xóa bản ghi theo chỉ số

47. Sử dụng hàm describe() để thống kê dữ liệu

48. Xem thống kê trên từng cột

49. Vẽ đồ thị xem phân bố giá trị của 1 trường trong dataframe

50. Tạo mới dataframe từ các python list

51. Sắp xếp dataframe

52. Nối 2 dataframe (Lỗi)

53. Xáo trộn các bản ghi trong dataframe

54. Lưu dataframe về file csv

55. Tối ưu bộ nhớ khi dùng pandas

56. Tạo 1 file chương trình hiện các menu gồm các mục trên và mục cuối là thoát. Người dùng chọn chức năng nào trong menu thì chương trình sẽ thực hiện chức năng tương ứng.

4) BÀI 4: Mini project: Xử lý dữ liệu bảng điểm bằng Pandas

Yêu cầu dự án: Cho trước các bộ dữ liệu sau

[dataset.zip](#)

Giải thích về bộ dữ liệu

- roster.csv: Chứa danh sách sinh viên, gồm các thông tin:
 - ID: ID của sinh viên
 - Name: Họ tên đầy đủ của sinh viên
 - Net ID: Net ID của sinh viên
 - Email Address: Địa chỉ mail của sinh viên
 - Section: Ca học của sinh viên
- hw_exam_grades.csv: Chứa thông tin về điểm của các bài tập về nhà và bài kiểm tra
 - First Name: Tên
 - Last Name: Họ
 - SID: SID của sinh viên
 - Homework x: Điểm của sinh viên trong bài tập thứ x
 - Homework x - Max Points: Điểm tối đa của bài tập thứ x
 - Homework x - Submission Time: Thời gian nộp bài tập của sinh viên
 - Exam x: Điểm của bài kiểm tra thứ x
 - Exam x - Max Points: Điểm tối đa của bài kiểm tra thứ x
 - Exam x - Submission Time: Thời gian nộp bài kiểm tra của sinh viên
- quiz_x_grades.csv: Chứa thông tin điểm trong quiz thứ x:
 - Last Name: Họ
 - First Name: Tên
 - Email: Địa chỉ mail của sinh viên
 - Grade: Điểm quiz của sinh viên

Yêu cầu

Hãy xử lý dữ liệu từ dữ liệu thô cho trước để xây dựng bảng điểm cho mỗi ca.

Kết quả xây dựng được như sau:

[result.zip](#)

Các nhiệm vụ

Nhiệm vụ 1

Hãy nhận xét về bộ dữ liệu.

Nhiệm vụ 2

1. Load dữ liệu roster.csv lên:
 - NetID và Email Address cần được chuyển thành định dạng viết thường.
 - Chỉ giữ lại cột Section, Email Address, NetID
 - Cột index là NetID
 - Gợi ý
Tham khảo tại liệu [Read CSV](#)
2. Load dữ liệu hw_exam_grades.csv lên:
 - SID cần được chuyển thành định dạng viết thường.
 - Bỏ đi các cột Submission
 - Cột index là SID
 - Gợi ý
Tham khảo tại liệu [Read CSV](#)
3. Load dữ liệu quiz_grades:
 - Gộp các bảng trong quiz_x_grades.csv lại thành một DataFrame
 - Email cần được chuyển thành định dạng viết thường.
 - Giữ lại cột Email và Grade
 - Cột index là Email
 - Đổi tên cột Grade thành Quiz 1, Quiz 2, ...
 - Gợi ý
Tham khảo tại liệu [Rename column of DataFrame](#) và [Concat DataFrame](#)

Nhiệm vụ 3

- Trộn 3 DataFrame có được từ nhiệm vụ 2 thành 1 DataFrame duy nhất
- Dữ liệu NaN được chuyển thành 0
- Gợi ý
Tham khảo tại liệu [DataFrame Merging](#)

Nhiệm vụ 4

1. Tính điểm bài kiểm tra: Lấy điểm kiểm tra chia cho điểm tối đa
 - Gợi ý
Tham khảo tại liệu [Calculating](#)
2. Tính điểm homework:

Có 2 cách tính điểm:

 - Theo tổng điểm: Tính tổng điểm thô và điểm tối đa một cách độc lập, sau đó lấy tỷ lệ.

- Theo điểm trung bình: Chia từng điểm thô cho số điểm tối đa tương ứng, sau đó lấy tổng của các tỷ lệ này và chia tổng cho số lượng bài tập.

Điểm sẽ được tính theo 2 cách, điểm sinh viên nhận được là điểm lớn nhất trong 2 điểm này.

 - Gợi ý
 - Tham khảo tại liệu [Filter with Regex](#) và [Max function](#)
 - 3. Tính điểm quiz theo cách tương tự homework

Điểm tối đa của mỗi quiz như sau:

 - Quiz 1: 11
 - Quiz 2: 15
 - Quiz 3: 17
 - Quiz 4: 14
 - Quiz 5: 12
 - 4. Tính điểm trung bình (final score)

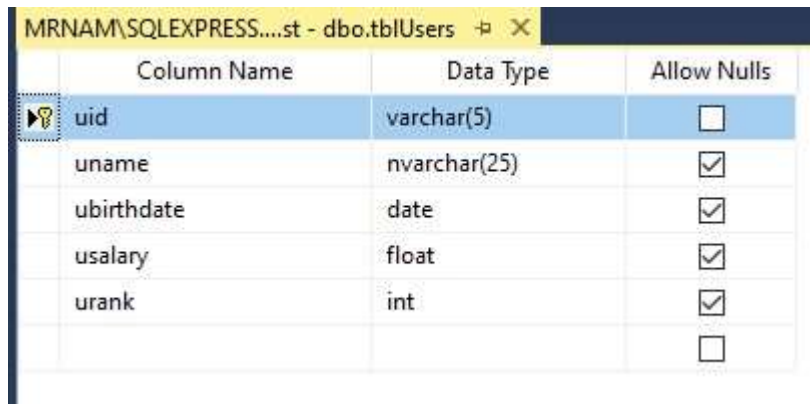
Trọng số các cột điểm như sau:

 - Exam 1: 0.05
 - Exam 2: 0.1
 - Exam 3: 0.15
 - Quiz Score: 0.3
 - Homework Score: 0.4

Điểm được làm tròn lên (ceiling)
 - 5. Tính điểm chữ

Điểm chữ được tính như sau:

 - Từ 90 điểm trở lên: A
 - Từ 80 đến cận 90: B
 - Từ 70 đến cận 80: C
 - Từ 60 đến cận 70: D
 - Dưới 60: F
- 5) Bài 5: Kết Nối Với Sql Server
- 1) Sinh viên tạo quyền truy cập MS-SQL Server
 - a. Tài khoản: sa
 - b. Mật khẩu: 123456
 - 2) Tạo CSDL mang tên dbPythonTestSQL
 - 3) Tạo bảng dữ liệu tblUsers như thiết kế



Column Name	Data Type	Allow Nulls
uid	varchar(5)	<input type="checkbox"/>
uname	nvarchar(25)	<input checked="" type="checkbox"/>
ubirthdate	date	<input checked="" type="checkbox"/>
usalary	float	<input checked="" type="checkbox"/>
urank	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

4) Nạp các dữ liệu mẫu

	uid	uname	ubirthdate	usalary	urank
	u0001	Henry	2001-07-31	125.5	1
	u0002	Peter	1990-08-25	500	1
	u0003	Owen	1995-02-28	275.5	2
	u0004	Jackson	2002-04-25	400	4
	u0005	Elite	2000-05-20	600	3

5) Đọc bài này: <https://www.educative.io/answers/how-to-connect-the-sql-server-withpython>

6) Tiến hành thực hiện các thao tác sau từ Python

- Lấy thông tin nhân viên có tuổi nhỏ hơn 35
- Lấy thông tin uname và tuổi (age) của tất cả các nhân viên
- Thêm nhân viên mới (u0006,"David","07-22-2000",450,5)
- Cập nhật lương của nhân viên u0002 thành 650
- Xóa nhân viên u0006

BÀI TẬP TUẦN 6: HỒI QUY TUYẾN TÍNH

1) Dữ liệu tuyensinhdaihoc

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#Đọc file dulieutuyensin
df =
pd.read_csv('../EDA/dulieuxettuyendaihoc.csv',header=0,delimiter=',',encoding='utf-8')
```

```
'''
Phân tích hồi quy tuyến tính
Mục đích: Phân tích tác động hay ảnh hưởng giữa các yếu tố đến mục tiêu (thường
đùng cho các biến (yếu tố) định lượng)
```

Thường vẽ biểu đồ Scatter để khám phá mối tương quan tuyến tính trước khi khám phá quan hệ hồi quy tuyến tính

PHƯƠNG PHÁP

1. Xác định biến độc lập (yếu tố) và biến phụ thuộc (mục tiêu)
2. Ghi ra phương trình hồi quy tuyến tính tổng quát $y = f(x)$
3. Chạy dữ liệu mô hình
4. Đọc các giá trị quan trọng và kết luận
5. Dự báo giá trị biến phụ thuộc khi biết trước giá trị biến độc lập

```
# Hãy cho biết sự ảnh hưởng của điểm học kì 1 năm lớp 12 đến điểm học kì 2 năm
lớp 12
```

```
#import statsmodels.api as sm
#pip install statsmodels -cài đặt thư viện này để sử dụng thư viện
#statsmodels.api
import statsmodels.api as sm
#linear regression
'''
```

1. Biến độc lập: học kì 1 (T5)
Biến phụ thuộc : học kì 2 (T6)
2. $T6 = f(T5)$
 $= A_0 + A_1 * T5 + \text{epsilon}$
3. Chạy mô hình
4. Đọc và hiểu kết quả

```
# adding a constant
X_with_constant = sm.add_constant(df[["T5"]].values)

y = df[["T6"]].values

# performing the regression
result = sm.OLS(y,X_with_constant).fit()

# Result of statsmodels
print(result.summary())
```

```
'''
Căn cứ vào điểm số T5 sẽ giải thích được 60% sự thay đổi của T6 (Adj. R-squared)
Prob (F-statistic) < 0.05: cho biết mô hình có khả năng phù hợp cho tổng thể [nó
là p-value]
const (2,11) chính là A0
x1 là A1
=> T6 = 2.113 + 0.7182 * T5

P>|t| = 0.000 rất nhỏ < 5% => x1 tương ứng với T5 (x` = T5) => biến T5 có ý nghĩa
thống kê trong phương trình này hay nói T5 có
ý nghĩa tham gia đánh giá tác động tới biến T6
'''

'''
Bước 5: Giả sử T5 = 7.5, dự báo T6 = 7.499
'''
```

```
# Khám phá sự ảnh hưởng của T6 đến điểm thi LOGIC
# adding a constant
X_with_constant = sm.add_constant(df[["T6"]].values)

y = df[["LOGIC"]].values

# performing the regression
result = sm.OLS(y,X_with_constant).fit()

# Result of statsmodels
print(result.summary())
'''
```

```
Adj. R-squared = 8%: quá ít, không thể giải thích cho điểm LOGIC, nói cách khác k
dựa T6 để gthich dc
Prob (F-statistic) = 0.00230 => phù hợp, bé hơn 0.05, có ý nghĩa thống kê
LOGIC = 2.6287 + 0.2344 * T6
P>|t| = 0.000 < 0.05: T6 có ý nghĩa thống kê
Giả sử T6 = 7.0
=> LOGIC = 4.2695
'''
```

```
# Khám phá sự ảnh hưởng của T6 đến điểm thi LOGIC
# adding a constant
X_with_constant = sm.add_constant(df[["T6"]].values)

y = df[["TOANLOGICPHANTICH"]].values

# performing the regression
result = sm.OLS(y,X_with_constant).fit()

# Result of statsmodels
print(result.summary())
'''
Adj. R-squared = 8%: quá ít, không thể giải thích cho điểm LOGIC, nói cách khác k
dựa T6 để gthich dc
Prob (F-statistic) = 0.00230 => phù hợp, bé hơn 0.05, có ý nghĩa thống kê
LOGIC = 2.6287 + 0.2344 * T6
P>|t| = 0.000 < 0.05: T6 có ý nghĩa thống kê
Giả sử T6 = 7.0
=> LOGIC = 4.2695
'''
```

```
X_with_constant = sm.add_constant(df[["T5","T6"]].values)

y = df[["TOANLOGICPHANTICH"]].values

# performing the regression
result = sm.OLS(y,X_with_constant).fit()

# Result of statsmodels
print(result.summary())
'''
1. Độc lập T5,T6
phụ thuộc Logic
2. Logic f(T5,T6)
```

```
Logic = A0 + A1*T5 + A2*T6 * epsilon = 2.7072 - 0.0913*T5 + 0.3115 * T6 + epsilon
'''
```

```
df = df
[['T5', 'T6', 'GT', 'DT', 'KV', 'KT', 'NGONNGU', 'TOANLOGICPHANTICH', 'GIAIQUYETVANDE', 'N
GAYTHI', 'DINHUUONGNGHENGHIEP']]
df.rename(columns={'TOANLOGICPHANTICH': 'LOGIC',
                  'GIAIQUYETVANDE': 'UNGXU',
                  'DINHUUONGNGHENGHIEP': 'HUONGNGHIEP'}, inplace=True)
```

```
# Hãy phân tích sự ảnh hưởng của điểm toán học kì 1,2 năm lớp 12 đến điểm NGONNGU
X_with_constant = sm.add_constant(df[["T5", "T6"]].values)

y = df[['NGONNGU']].values

# performing the regression
result = sm.OLS(y, X_with_constant).fit()

# Result of statsmodels
print(result.summary())
'''
Adj. R-squared = 1% : quá ít, k gthich dc gì
Prob (F-statistic): 7%
'''
```

```
# Đánh giá mức độ tác động giữa các yếu tố đến 1 đối tượng bằng phân tích hồi quy
tuyến tính

# Hãy cho biết mức độ tác động của T5, T6 (độc lập) đến điểm LOGIC (phụ thuộc)
# adding a constant
X = df[["T5", "T6"]].values

y = df[['LOGIC']].values

# performing the regresssion
result = sm.OLS(y, X).fit()

# result of statsmodels
print(result.summary())
'''
x2 = |0.5934| => T6 tác động mạnh hơn so với T5 (x1 = |0.0063|)
vì x2 dương nên tác động tích cực, còn âm mới tác động tiêu cực (nghịch biến)
```

```
'''
# Đánh giá mức độ tác động giữa các yếu tố đến 1 đối tượng bằng phân tích hồi quy
tuyến tính

# Hãy cho biết mức độ tác động của T5, T6 đến điểm UNGXU
# adding a constant
X = df[["T5","T6"]].values

y = df[['UNGXU']].values

# performing the regresssion
result = sm.OLS(y,X).fit()

# result of statsmodels
print(result.summary())

'''

x2 = |0.5318| => T6 tác động mạnh hơn so với T5 (x1 = |0.1519|)
vì x2 dương nên tác động tích cực (đồng biến), còn âm mới tác động tiêu cực
(ngược biến)
'''
```

Đọc thêm: Linear Regression in Python using Statsmodels

Linear regression analysis is a statistical technique for predicting the value of one variable(dependent variable) based on the value of another(independent variable). The dependent variable is the variable that we want to predict or forecast. In simple linear regression, there's one independent variable used to predict a single dependent variable. In the case of multilinear regression, there's more than one independent variable. The independent variable is the one you're using to forecast the value of the other variable. The **statsmodels.regression.linear_model.OLS** method is used to perform linear regression. Linear equations are of the form:

Syntax: *statsmodels.regression.linear_model.OLS(endog, exog=None, missing='none', hasconst=None, **kwargs)*

Parameters:

- **endog:** array like object.
- **exog:** array like object.
- **missing:** str. None, decrease, and raise are the available alternatives. If the value is 'none,' no nan testing is performed. Any observations with nans are dropped if 'drop' is selected. An error is raised if 'raise' is used. 'none' is the default.

- **hasconst: None or Bool.** Indicates whether a user-supplied constant is included in the RHS. If True, k constant is set to 1 and all outcome statistics are calculated as if a constant is present. If False, k constant is set to 0 and no constant is verified.
- ****kwargs:** When using the formula interface, additional arguments are utilised to set model characteristics.

Return: Ordinary least squares are returned.

Installation

```
pip install numpy
pip install pandas
pip install statsmodels
```

Stepwise Implementation

Step 1: Import packages.

Importing the required packages is the first step of modeling. The pandas, NumPy, and stats model packages are imported.

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
```

Step 2: Loading data.

To access the CSV file click [here](#). The CSV file is read using `pandas.read_csv()` method. The head or the first five rows of the dataset is returned by using the `head()` method. Head size and Brain weight are the columns.

- Python3

```
df = pd.read_csv('headbrain1.csv')
df.head()
```

The head of the data frame looks like this:

Out[2]:

	Head Size(cm ³)	Brain Weight(grams)
0	4512	1530
1	3738	1297
2	4261	1335
3	3777	1282
4	4177	1590

Visualizing the data:

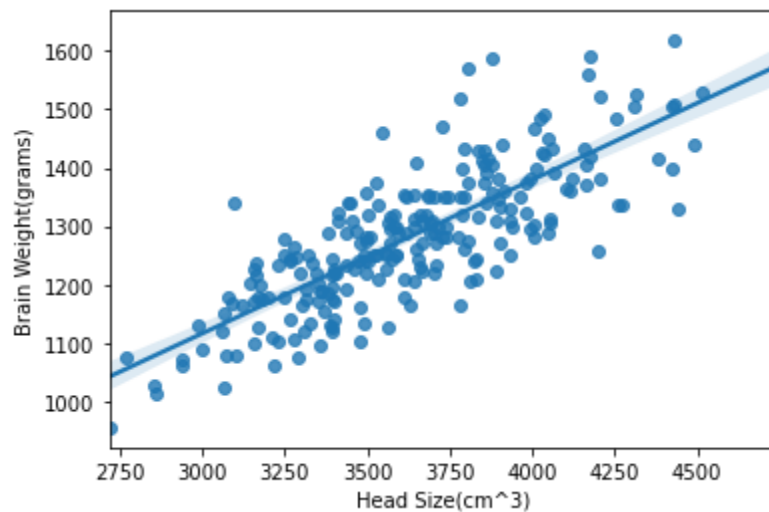
By using the [matplotlib](#) and seaborn packages, we visualize the data. [sns.regplot\(\)](#) function helps us create a regression plot.

- Python3

```
# import packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('headbrain1.csv')
sns.regplot('Head Size(cm^3)', 'Brain
Weight(grams)', data=df)
plt.show()
```

Output:



Step 3: Setting a hypothesis.

- Null hypothesis (H0): There is no relationship between head size and brain weight.
- Alternative hypothesis (Ha): There is a relationship between head size and brain weight.

Step 4: Fitting the model

[statsmodels.regression.linear_model.OLS\(\)](#) method is used to get ordinary least squares, and `fit()` method is used to fit the data in it. The `ols` method takes in the data and performs linear regression. we provide the dependent and independent columns in this format :

independent_columns ~ dependent_column:

left side of the ~ operator contains the independent variables and right side of the operator contains the name of the dependent variable or the predicted column.

• Python3

```
df.columns = ['Head_size', 'Brain_weight']
model = smf.ols(formula='Head_size ~ Brain_weight',
data=df).fit()
```

Step 5: Summary of the model.

All the summary statistics of the linear regression model are returned by the `model.summary()` method. The p-value and many other values/statistics are known by this method. Predictions about the data are found by the `model.summary()` method.

```
print(model.summary())
```

Code Implementation:

```
# import packages
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
# loading the csv file
df = pd.read_csv('headbrain1.csv')
print(df.head())
# fitting the model
df.columns = ['Head_size', 'Brain_weight']
model = smf.ols(formula='Head_size ~ Brain_weight',
data=df).fit()
# model summary
print(model.summary())
```

Output:

```

=====
                        OLS Regression Results
=====
Dep. Variable:          Head_size      R-squared:                0.639
Model:                  OLS            Adj. R-squared:          0.638
Method:                 Least Squares   F-statistic:             416.5
Date:                   Sun, 08 May 2022 Prob (F-statistic):       5.96e-54
Time:                   21:40:40        Log-Likelihood:          -1613.4
No. Observations:       237            AIC:                     3231.
Df Residuals:           235            BIC:                     3238.
Df Model:                1
Covariance Type:        nonrobust
=====
                        coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept              520.6101    153.215      3.398      0.001     218.759     822.461
Brain_weight           2.4269       0.119     20.409      0.000       2.193       2.661
=====
Omnibus:                2.687    Durbin-Watson:           1.726
Prob(Omnibus):           0.261    Jarque-Bera (JB):        2.321
Skew:                    0.207    Prob(JB):                 0.313
Kurtosis:                3.252    Cond. No.                 1.38e+04
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.38e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

```

Description of some of the terms in the table :

- **R- squared value:** R-squared value ranges between 0 and 1. An R-squared of 100 percent indicates that all changes in the dependent variable are completely explained by changes in the independent variable(s). if we get 1 as an r-squared value it means there's a perfect fit. In our example, the r-squared value is 0.638.
- **F- statistic:** The F statistic simply compares the combined effect of all variables. In simplest terms, reject the null hypothesis if your alpha level is greater than your p-value.
- **coef:** the coefficients of the independent variables in the regression equation.

Our predictions:

If we take our significance level (alpha) to be 0.05, we reject the null hypothesis and accept the alternative hypothesis as $p < 0.05$. so, we can say that there is a relationship between head size and brain weight.

Hướng dẫn summary of linear regression in python - tóm tắt hồi quy tuyến tính trong python

Thường thì bạn có thể muốn trích xuất một bản tóm tắt của một mô hình hồi quy được tạo bằng cách sử dụng Scikit-learn trong Python.

Nội dung chính [Show](#)

Thật không may, Scikit-Learn không cung cấp nhiều chức năng tích hợp để phân tích bản tóm tắt của mô hình hồi quy vì nó thường chỉ được sử dụng cho mục đích dự đoán.

Vì vậy, nếu bạn quan tâm đến việc có được một bản tóm tắt về mô hình hồi quy trong Python, bạn có hai tùy chọn:

1. Sử dụng các chức năng hạn chế từ scikit-learn. Use limited functions from scikit-learn.

2. Sử dụng StatModels thay thế. Use statsmodels instead.

Các ví dụ sau đây cho thấy cách sử dụng từng phương thức trong thực tế với các cấu trúc sau đây:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'x1': [1, 2, 2, 4, 2, 1, 5, 4, 2, 4, 4],
                  'x2': [1, 3, 3, 5, 2, 2, 1, 1, 0, 3, 4],
                  'y': [76, 78, 85, 88, 72, 69, 94, 94, 88, 92, 90]})

#view first five rows of DataFrame
df.head()
```

	x1	x2	y
0	1	1	76
1	2	3	78
2	2	3	85
3	4	5	88
4	2	2	72

Phương pháp 1: Nhận bản tóm tắt mô hình hồi quy từ Scikit-learn

Chúng ta có thể sử dụng mã sau để phù hợp với mô hình hồi quy tuyến tính nhiều bằng cách sử dụng Scikit-LEARN:

```
from sklearn.linear_model import LinearRegression

#initiate linear regression model
model = LinearRegression()

#define predictor and response variables
X, y = df[['x1', 'x2']], df.y

#fit regression model
model.fit(X, y)
```

Sau đó, chúng ta có thể sử dụng mã sau để trích xuất các hệ số hồi quy của mô hình cùng với giá trị R-Squared của mô hình:

```
#display regression coefficients and R-squared value of model
print(model.intercept_, model.coef_, model.score(X, y))
```

```
70.4828205704 [ 5.7945 -1.1576] 0.766742556527
```

Sử dụng đầu ra này, chúng ta có thể viết phương trình cho mô hình hồi quy được trang bị:

$$y = 70,48 + 5,79x_1 - 1,16x_2$$

Chúng ta cũng có thể thấy rằng giá trị R² của mô hình là 76,67. & NBSP;

Điều này có nghĩa là & nbsp; 76,67% biến thể trong biến phản hồi có thể được giải thích bằng hai biến dự đoán trong mô hình. **76.67%** of the variation in the response variable can be explained by the two predictor variables in the model.

Mặc dù đầu ra này rất hữu ích, nhưng chúng tôi vẫn không biết tổng thể F-thống kê & nbsp; của mô hình, giá trị p của các hệ số hồi quy cá nhân và các số liệu hữu ích khác có thể giúp chúng tôi hiểu mô hình phù hợp với bộ dữ liệu này như thế nào.

Phương pháp 2: Nhận tóm tắt mô hình hồi quy từ StatModels

Nếu bạn quan tâm đến việc trích xuất bản tóm tắt mô hình hồi quy trong Python, thì bạn nên sử dụng gói StatModels **statsmodels** package.

Mã sau đây cho thấy cách sử dụng gói này để phù hợp với cùng một mô hình hồi quy tuyến tính giống như ví dụ trước và trích xuất bản tóm tắt mô hình:

```
import statsmodels.api as sm

#define response variable
y = df['y']

#define predictor variables
x = df[['x1', 'x2']]

#add constant to predictor variables
x = sm.add_constant(x)

#fit linear regression model
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

OLS Regression Results

```
=====
```

```

Dep. Variable:          y      R-squared:          0.767
Model:                  OLS    Adj. R-squared:       0.708
Method:                 Least Squares    F-statistic:         13.15
Date:                   Fri, 01 Apr 2022    Prob (F-statistic):   0.00296
Time:                   11:10:16    Log-Likelihood:      -31.191
No. Observations:       11    AIC:                 68.38
Df Residuals:           8    BIC:                 69.57
Df Model:                2
Covariance Type:        nonrobust

```

```

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
const          70.4828        3.749      18.803      0.000      61.839      79.127
x1              5.7945        1.132       5.120      0.001       3.185       8.404
x2             -1.1576        1.065      -1.087      0.309     -3.613       1.298
=====
Omnibus:                0.198    Durbin-Watson:          1.240
Prob(Omnibus):           0.906    Jarque-Bera (JB):         0.296
Skew:                   -0.242    Prob(JB):                 0.862
Kurtosis:                2.359    Cond. No.                 10.7
=====

```

Lưu ý rằng các hệ số hồi quy và giá trị bình phương R phù hợp với các hệ số được tính toán bởi Scikit-learn, nhưng chúng tôi cũng cung cấp một tấn các số liệu hữu ích khác cho mô hình hồi quy. Ví dụ: chúng ta có thể thấy các giá trị p cho từng biến dự đoán riêng lẻ:

- Giá trị p cho x1 = .001
- Giá trị P cho x2 = 0,309

Chúng ta cũng có thể thấy thống kê F tổng thể của mô hình, giá trị bình phương R được điều chỉnh, giá trị AIC của mô hình và nhiều hơn nữa.

Tài nguyên bổ sung

Các hướng dẫn sau đây giải thích cách thực hiện các hoạt động phổ biến khác trong Python:

Cách thực hiện hồi quy tuyến tính đơn giản trong Python Cách thực hiện hồi quy tuyến tính nhiều trong Python Cách tính AIC của các mô hình hồi quy trong Python

How to Perform Multiple Linear Regression in Python

How to Calculate AIC of Regression Models in Python

Tóm tắt hồi quy tuyến tính là gì?

Hồi quy tuyến tính là gì? Hồi quy tuyến tính là một mô hình thống kê phân tích mối quan hệ giữa một biến phản hồi (thường được gọi là y) và một hoặc nhiều biến và tương tác của chúng (thường được gọi là X hoặc

các biến giải thích). a statistical model that analyzes the relationship between a response variable (often called y) and one or more variables and their interactions (often called x or explanatory variables).

Hồi quy tuyến tính trong Python là gì?

Hồi quy tuyến tính là một phương pháp thống kê để mô hình hóa mối quan hệ giữa một biến phụ thuộc với một tập hợp các biến độc lập nhất định. Lưu ý: Trong bài viết này, chúng tôi gọi các biến phụ thuộc là phản hồi và các biến độc lập là các tính năng để đơn giản. a statistical method for modeling relationships between a dependent variable with a given set of independent variables. Note: In this article, we refer to dependent variables as responses and independent variables as features for simplicity.

Làm thế nào để bạn tìm thấy bản tóm tắt của một mô hình trong Python?

Nếu bạn muốn trích xuất bản tóm tắt mô hình hồi quy trong Python, bạn nên sử dụng gói StatModels. use the statsmodels package.

Hồi quy tuyến tính giải thích với ví dụ là gì?

Hồi quy tuyến tính thường được sử dụng để phân tích và mô hình hóa dự đoán. Ví dụ, nó có thể được sử dụng để định lượng các tác động tương đối của tuổi, giới tính và chế độ ăn uống (các biến dự đoán) về chiều cao (biến kết quả). commonly used for predictive analysis and modeling. For example, it can be used to quantify the relative impacts of age, gender, and diet (the predictor variables) on height (the outcome variable).

Hướng dẫn linear regression python residuals - phần dư python hồi quy tuyến tính

Trong hướng dẫn này, bạn sẽ thấy cách thực hiện hồi quy tuyến tính nhiều trong Python bằng cả Sklearn và StatModels.

Nội phân chính

- Kiểm tra tuyến tính
- Thực hiện hồi quy tuyến tính nhiều
- (1) Phần đầu tiên hiển thị đầu ra được tạo bởi sklearn:
- (2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.

Dưới đây là các chủ đề được đề cập:

- Xem xét ví dụ sẽ được sử dụng trong hướng dẫn này
- Kiểm tra tuyến tính
- Thực hiện hồi quy tuyến tính nhiều

(1) Phần đầu tiên hiển thị đầu ra được tạo bởi sklearn:

- (2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.
- unemployment_rate

```
import pandas as pd
```



```
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,20
16,2016,2016,2016,2016,2016,2016],
'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.
75,1.75,1.75,1.75,1.75],
'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,
6.2,6.2,6.1],
'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,9
49,884,866,876,822,704,719]
}
df = pd.DataFrame(data)
print(df)
```

Thực hiện hồi quy tuyến tính nhiều trong Python

Trong ví dụ sau, chúng tôi sẽ thực hiện hồi quy tuyến tính cho một nền kinh tế hư cấu, trong đó index_price là biến phụ thuộc và 2 biến độc lập/đầu vào là:

1) Kiểm tra tuyến tính

Thực hiện & nbsp; hồi quy tuyến tính nhiều

(1) Phần đầu tiên hiển thị đầu ra được tạo bởi & nbsp; sklearn:

(2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.

- Dưới đây là các chủ đề được đề cập:
- Xem xét ví dụ sẽ được sử dụng trong hướng dẫn này

Thực hiện hồi quy tuyến tính nhiều trong Python

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,20
16,2016,2016,2016,2016,2016,2016],
'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.
75,1.75,1.75,1.75,1.75],
'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,
6.2,6.2,6.1],
```

```

        'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,9
49,884,866,876,822,704,719]
    }

df = pd.DataFrame(data)

plt.scatter(df['interest_rate'], df['index_price'], color='red')
plt.title('Index Price Vs Interest Rate', fontsize=14)
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Index Price', fontsize=14)
plt.grid(True)
plt.show()

```

Trong ví dụ sau, chúng tôi sẽ thực hiện hồi quy tuyến tính cho một nền kinh tế hư cấu, trong đó index_price là biến phụ thuộc và 2 biến độc lập/đầu vào là:

```

import pandas as pd
import matplotlib.pyplot as plt
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,20
16,2016,2016,2016,2016,2016,2016,2016],
        'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
        'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.
75,1.75,1.75,1.75],
        'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,
6.2,6.2,6.1],
        'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,9
49,884,866,876,822,704,719]
    }
df = pd.DataFrame(data)
plt.scatter(df['unemployment_rate'], df['index_price'], color='green')
plt.title('Index Price Vs Unemployment Rate', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Index Price', fontsize=14)
plt.grid(True)
plt.show()

```

Xin lưu ý rằng bạn sẽ phải xác nhận rằng một số giả định được đáp ứng trước khi bạn áp dụng các mô hình hồi quy tuyến tính. Đáng chú ý nhất, bạn phải đảm bảo rằng mối quan hệ tuyến tính tồn tại giữa biến phụ thuộc và biến/s độc lập (nhiều hơn về phần kiểm tra tuyến tính).

Bây giờ, hãy nhảy vào bộ dữ liệu mà chúng tôi sẽ sử dụng. Dữ liệu sẽ được ghi lại & NBSP; sử dụng gấu trúc DataFrame:

2) Thực hiện hồi quy tuyến tính nhiều

(1) Phần đầu tiên hiển thị đầu ra được tạo bởi sklearn:

(2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi Statsmodels.

Dưới đây là các chủ đề được đề cập:

```
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,20
16,2016,2016,2016,2016,2016,2016,2016,2016],
      'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
      'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.
75,1.75,1.75,1.75,1.75],
      'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,6.1,5.9,
6.2,6.2,6.1],
      'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,9
49,884,866,876,822,704,719]
}
df = pd.DataFrame(data)
x = df[['interest_rate','unemployment_rate']]
y = df['index_price']
# with sklearn
regr = linear_model.LinearRegression()
regr.fit(x, y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
# with statsmodels
x = sm.add_constant(x) # adding a constant
model = sm.OLS(y, x).fit()
predictions = model.predict(x)
print_model = model.summary()
```

```
print(print_model)
```

Xem xét ví dụ sẽ được sử dụng trong hướng dẫn này

(1) Phần đầu tiên hiển thị đầu ra được tạo bởi & nbsp; sklearn:

```
Intercept:
1798.4039776258564
Coefficients:
[ 345.54008701 -250.14657137]
```

(2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.

Dưới đây là các chủ đề được đề cập: $\text{intercept} + (\text{interest_rate coef}) * X1 + (\text{unemployment_rate coef}) * X2$

Xem xét ví dụ sẽ được sử dụng trong hướng dẫn này

Thực hiện hồi quy tuyến tính nhiều trong Python $1798.4040 + (345.5401) * X1 + (-250.1466) * X2$

(2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.

Dưới đây là các chủ đề được đề cập:

OLS Regression Results						
=====						
Dep. Variable:	index_price	R-squared:	0.898			
Model:	OLS	Adj. R-squared:	0.888			
Method:	Least Squares	F-statistic:	92.07			
Date:	Sat, 30 Jul 2022	Prob (F-statistic):	4.04e-11			
Time:	13:47:01	Log-Likelihood:	-134.61			
No. Observations:	24	AIC:	275.2			
Df Residuals:	21	BIC:	278.8			
Df Model:	2					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

const	1798.4040	899.248	2.000	0.059	-71.685	3668.493
interest_rate	345.5401	111.367	3.103	0.005	113.940	577.140
unemployment_rate	-250.1466	117.950	-2.121	0.046	-495.437	-4.856
=====						
Omnibus:	2.691	Durbin-Watson:	0.530			
Prob(Omnibus):	0.260	Jarque-Bera (JB):	1.551			
Skew:	-0.612	Prob(JB):	0.461			
Kurtosis:	3.226	Cond. No.	394.			
=====						

Xem xét ví dụ sẽ được sử dụng trong hướng dẫn này

Thực hiện hồi quy tuyến tính nhiều trong Python

- 3) Trong ví dụ sau, chúng tôi sẽ thực hiện hồi quy tuyến tính cho một nền kinh tế hư cấu, trong đó `index_price` là biến phụ thuộc và 2 biến độc lập/đầu vào là:

Xin lưu ý rằng bạn sẽ phải xác nhận rằng một số giả định được đáp ứng trước khi bạn áp dụng các mô hình hồi quy tuyến tính. Đáng chú ý nhất, bạn phải đảm bảo rằng mối quan hệ tuyến tính tồn tại giữa biến phụ thuộc và biến/s độc lập (nhiều hơn về phần kiểm tra tuyến tính).

<https://v1study.com/python-tham-khao-hoi-quy-tuyen-tinh-trong-python.html>

2) Các Lab