BÀI TẬP THỰC HÀNH PHÂN TÍCH DỮ LIỆU 1

BÀI TẬP TUẦN 1

1) ÔN PYTHON
2) DAY 1
3) DAY 2
4) TUẦN 1
5) Các bài Lab: Lab-01-OnPython.pdf

BÀI TẬP TUẦN 2

1) Lab 1: Pandas cơ bản

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random

Ô chứa mã <undefined>
# %% [code]
#1. Đọc File với các file trong thư mục dữ liệu. Ứng với mỗi file sinh viên thực hiện các câu sau
df =pd.read_csv('orginal_sales_data_edit.csv')
df
```

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

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

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

```
#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 10
df['PRODUCTLINE']
```

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

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

```
#14. loai bo cac dong trung nhau
df.drop_duplicates(inplace=True)
df
```

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

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

```
#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

```

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


```
#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 sales 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]
print('Cách 2')
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_map1 ={1:'Qui_1',2:'Qui_2',3:'Qui_3',4:'Qui_4'}
df['QTR_ID']=df['QTR_ID'].map(dict_map1)
df['QTR_ID']
```

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

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

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

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

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

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

```
#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.
df['DaGiao'] =df['STATUS']=='Shipped'
df.head(5)
```

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

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

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

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

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

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

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

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

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

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

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

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

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

```python
#55.  Tối ưu bộ nhớ khi dùng pandas
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the data type
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object and col_type.name != 'category' and 'datetime' not in col_type.name:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        elif 'datetime' not in col_type.name:
            df[col] = df[col].astype('category')
```

```python
        elif 'datetime' not in col_type.name:
            df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df
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 dont 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.
#lmplot() 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.lmplot(x, y, data, hue=None, col=None, row=None, **kwargs)
# loading dataset
data = sns.load_dataset("tips")
sns.lmplot(x='total_bill', y='tip', data=data)
plt.show()
```

```
#regplot() method is also similar to lmplot 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: 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 ANALYSYS============
#Đây lá 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('orginal_sales_data_edit.csv')
df =pd.read_csv('orginal_sales_data_edit.csv',encoding='utf-8',
header=0,delimiter=',')
df
#Kết quả: 2824 dòng
```

```
#1.2. Loại bỏ dòng dữ liệu rỗng
df.dropna(how='all',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
```

```
#1.3. 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 vời Seaborn Displt
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('orginal_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 BIET 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=su
m)
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 BIẾT XÁC SUẤT XẢY RA TRÊN MỖI ĐOẠN
#4.2. VE BIEU DO HIS CUA QUANTITYORDERED THEO DEALISIZE //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 DEALISIZE //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 DEALISIZE //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. VE BIEU DO PAIRPLOT CUA QUANTITYORDERED,PRICEEACH,SALES
sns.pairplot(df[['QUANTITYORDERED','PRICEEACH','SALES']],diag_kind='hist',kind='kde')
plt.show()
```

```
BIỂU DIỄN TƯƠNG QUAN BIẾN SỐ THEO TỪNG CẶP
```

```
#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
#BIEU DIEN (BOXPLOT,VIOLIN) CUA QUANTITYODERED TREN CUNG CHART
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title('Simple plot')
```

```
#fig, axs = 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

```
BIEU DIEN BOXPLOT CUA QUANTITYORDERED THEONHOM DEALSIZE
```

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

```
BIEU DIENCATPLOT CUA QUANTITYORDERED THEO NHOM DEALSIZE
sns.catplot(x='YEAR_ID',y='QUANTITYORDERED',hue='DEALSIZE',data=df)
plt.show()
```

```
BIEU DIEN BOXPLOT CUA QUANTITYORDERED THEO NAM,NHOM 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 (KURTORIS) CUA QUANTITYOREDERED , 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=Fa
lse).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 lệnh chuẩn của SALES theo từng nhóm DEALSIZE
df.groupby(['DEALSIZE'])['SALES'].agg(['sum','mean','std'])
```

```
#Thống kê tổng, trung vị, độ lệnh 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: tháo 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
```

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

```
# Đổi tên cột
df.rename(columns={'TOANLOGICPHANTICH':'LOGIC','GIAIQUYETVANDE':'UNGXU',
'DINHHUONGNGHENGHIEP':'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['TBTOAN'] = (df['T5']+df['T6'])/2
df.head(5)
```

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

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

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

df.loc[(df['TBTOAN'] >= 9.0), 'XEPLOAI'] = 'EXCEL'
```

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

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

```
'''
Tạo biến nhóm khối thi NHOMKT thõa 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 TBTOAN >= 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['TBTOAN']))
```

```
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()
```

```
'''
KHi 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 thitrên 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 (Confficient 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 xuất
Làm mịn với phân phối xác xuất
'''
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 nheièu 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 tren', 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 vien 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
```

```
'''
Skewess = 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 xuất được phân bổ lệch về phía nào nhiều
Trị tuyệt đối giá trị càng lớn thì dữ liệu phân phối nghiên 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 tị 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 Kutorsis tức là đã chuẩn hóa giá trị về excess
kutorsis theo mean = 0

+ Trị tuyệt đối excess kutorsis 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 kutorsis 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,squ
are=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()
```