```
#importing
from google.colab import drive
drive.mount('/content/gdrive')

import pandas as pd
df=pd.read_csv('/content/Obesity_dataset.csv')
```

```
#EDITED DATASET TO REPLACE DECIMALS AND NUMBERS WITH THE ACTUAL ANSWERS FOR
ORDINAL DATA
df=pd.read_csv('/content/Obesity_dataset.csv')
for i in df['FCVC']:
  if i>=0 and i<1.5:
    df['FCVC']=df['FCVC'].replace(i,'Never')
  elif i>=1.5 and i<2.5:
    df['FCVC']=df['FCVC'].replace(i,'Sometimes')
  else:
    df['FCVC']=df['FCVC'].replace(i,'Always')


for i in df['CH2O']:
  if i>=0 and i<1.5:
    df['CH2O']=df['CH2O'].replace(i,'Less than 1L')
  elif i>=1.5 and i<2.5:
    df['CH2O']=df['CH2O'].replace(i,'Between 1-2L')
  else:
    df['CH2O']=df['CH2O'].replace(i,'More than 2L')


for i in df['FAF']:
  if i>=0 and i<0.5:
    df['FAF']=df['FAF'].replace(i,'0')
  elif i>=0.5 and i<1.5:
    df['FAF']=df['FAF'].replace(i,'1 to 2')
  elif i>=1.5 and i<2.5:
    df['FAF']=df['FAF'].replace(i,'2 to 4')
  else:
    df['FAF']=df['FAF'].replace(i,'4 to 5')
```

```
for i in df['TUE']:
  if i>=0 and i<0.5:
    df['TUE']=df['TUE'].replace(i,'0 to 2')
  elif i>=0.5 and i<1.5:
    df['TUE']=df['TUE'].replace(i,'3 to 5')
  else:
    df['TUE']=df['TUE'].replace(i,'>5')


for i in df['NCP']:
  if i>=0 and i<1.5:
    df['NCP']=df['NCP'].replace(i,'1')
  elif i>=1.5 and i<2.5:
    df['NCP']=df['NCP'].replace(i,'2')
  elif i>=2.5 and i<3.5:
    df['NCP']=df['NCP'].replace(i,'3')
  else:
    df['NCP']=df['NCP'].replace(i,'4')

df_copy = df.copy()
print(df)
```

```
#GENERALISED LISTS
import seaborn as sns
variables = ['Gender',
             'Age',
             'Height',
             'Weight',
             'Family History',
             'High Caloric Food Frequency',
             'Frequency of Vegetable Consumption',
             'Number of main meals',
             'Consumption of Food Between Meals',
             'Smoke',
             'Consumption of Water Daily',
             'Calories Consumption Monitoring',
             'Physical Activity Frequency',
             'Time Using Technology Devices',
             'Consumption of Alcohol',
             'Transportation Used',
             'Obesity Status']
```

```
shorthand =
['Gender','Age','Height','Weight','family_history_with_overweight','FAVC','FC
VC','NCP','CAEC','SMOKE','CH2O','SCC','FAF','TUE','CALC','MTRANS','NObeyesdad
']
continuous = [1,2,3]
ordinal = [6,8,10,12,13,14]
nominal = [0,4,5,9,11]
nominal_3 = [7,15]
order=['Insufficient_Weight',
       'Normal_Weight',
       'Overweight_Level_I',
       'Overweight_Level_II',
       'Obesity_Type_I',
       'Obesity_Type_II',
       'Obesity_Type_III']
```

```
#SPEARMAN + BOXPLOT FOR CONTINUOUS
corrs = []
p_values = []
for i in continuous:
  #GRAPH
  graph = sns.catplot(y=shorthand[i], x='NObeyesdad', data=df, kind='box',
order=order)
  graph.set(xlabel="Obesity Status", ylabel=variables[i])
  plt.xticks(rotation=90)
  #SPEARMAN
  corr, p_value = spearmanr(df[shorthand[i]], np.array([order.index(i) for i
in df['NObeyesdad']]))
  corrs.append(corr)
  p_values.append(p_value)
continuous_result = pd.DataFrame({'Variable':[variables[i] for i in
continuous],'Shorthand':[ shorthand[i] for i in continuous],'corr':corrs,'p
value':p_values})
print(continuous_result)
df=df_copy.copy()
```

```
#SPEARMAN + HEAT MAP FOR ORDINAL
import matplotlib.pyplot as plt
corrs = []
p_values = []
ordinal_order = [0,1,2,3,4,5,['Never','Sometimes','Always'],
```

```
                ['1','2','3','4'],
                ['no','Sometimes','Frequently','Always'],9,
                ['Less than 1L','Between 1-2L','More than 2L'],11,
                ['0','1 to 2','2 to 4', '4 to 5'],
                ['0 to 2','3 to 5','>5'],
                ['no','Sometimes','Frequently','Always']]


for i in ordinal:
  tempdf = pd.DataFrame({variables[i] :
pd.Categorical(df[shorthand[i]],categories=ordinal_order[i], ordered=True),
                        'Obesity Status' :
pd.Categorical(df['NObeyesdad'],categories=order,ordered=True)})
  #graph
  tempct = pd.crosstab(tempdf['Obesity Status'],tempdf[variables[i]])
  sns.heatmap(tempct, cmap="YlGnBu", annot=True, fmt='d').invert_yaxis()
  plt.ylabel('Obesity Status')
  plt.xlabel(variables[i])
  plt.show()
  #spearman
  corr, p_value = spearmanr(np.array([ordinal_order[i].index(x) for x in
tempdf[variables[i]]]), np.array([order.index(x) for x in tempdf['Obesity
Status']]))
  corrs.append(corr)
  p_values.append(p_value)
ordinal_result = pd.DataFrame({'Variable':[variables[i] for i in
ordinal],'Shorthand':[ shorthand[i] for i in ordinal],'corr':corrs,'p
value':p_values})
print(ordinal_result)
df=df_copy.copy()
```

```
#MANNWHITNEY + BARPLOT FOR NOMINAL (2 GROUPS)
import scipy.stats as stats
Us = []
p_values = []
category_map = {'Insufficient_Weight':1,
                'Normal_Weight':2,
                'Overweight_Level_I':3,
                'Overweight_Level_II':4,
                'Obesity_Type_I':5,
                'Obesity_Type_II':6,
```

```
                'Obesity_Type_III':7}
for i in nominal:
  #GRAPH
  sns.boxplot(y= df['NObeyesdad'].map(category_map), x= df[shorthand[i]])
  plt.ylabel('Obesity Status')
  plt.yticks([1,2,3,4,5,6,7], order)
  plt.xlabel(variables[i])
  plt.show()
  #MANNWHITNEY
  df['NObeyesdad'] = pd.Categorical(df['NObeyesdad'], categories=order)
  df['NObeyesdad'] = df['NObeyesdad'].cat.codes
  data = df[[shorthand[i] for i in nominal] + ['NObeyesdad']]
  groups = data.groupby(shorthand[i])['NObeyesdad'].apply(list).values
  U, p_value = mannwhitneyu(groups[0], groups[1])
  Us.append(U)
  p_values.append(p_value)
  df=df_copy.copy()
nominal_result = pd.DataFrame({'Variable':[variables[i] for i in
nominal],'Shorthand':[ shorthand[i] for i in nominal],'U':Us,'p
value':p_values})
print(nominal_result)
df=df_copy.copy()
```

```
#KRUSKAL-WALLIS + BAR PLOT FOR NOMINAL (>2 GROUPS)
from scipy import stats
Hs=[]
pvalues=[]

Nominal = ['MTRANS', 'NCP']
data = df[Nominal + ['NObeyesdad']]

groups = data.groupby('MTRANS')['NObeyesdad'].apply(list).values
H, pvalue = stats.kruskal(groups[0], groups[1], groups[2], groups[3],
groups[4])
Hs.append(H)
pvalues.append(pvalue)
groups = data.groupby('NCP')['NObeyesdad'].apply(list).values
H, pvalue = stats.kruskal(groups[0], groups[1], groups[2], groups[3])
Hs.append(H)
pvalues.append(pvalue)
```

```
nominal_3_result = pd.DataFrame({'Variable':[variables[i] for i in
nominal_3],'Shorthand':[ shorthand[i] for i in nominal_3],'H':Hs,'p
value':pvalues})
print(nominal_3_result)

#graph
sns.boxplot(y= df['NObeyesdad'].map(category_map), x= df['MTRANS'])
plt.ylabel('Obesity Status')
plt.yticks([1,2,3,4,5,6,7], order)
plt.xlabel('MTRANS')
plt.show()

sns.boxplot(y= df['NObeyesdad'].map(category_map), x=
df['NCP'],order=['1','2','3','4'])
plt.ylabel('Obesity Status')
plt.yticks([1,2,3,4,5,6,7], order)
plt.xlabel('NCP')
plt.show()
df=df_copy.copy()
```

```
from tabulate import tabulate

print('CONTINUOUS (SPEARMAN):')
print(tabulate(continuous_result.sort_values(by=['corr'],ascending=False),hea
ders=list(continuous_result.columns.values),tablefmt = 'fancy_grid'))
print('ORDINAL (SPEARMAN):')
print(tabulate(ordinal_result.sort_values(by=['corr'],key=abs,ascending=False
),headers=list(ordinal_result.columns.values),tablefmt = 'fancy_grid'))
print('NOMINAL (2 GROUPS) (MANNWHITNEYU):')
print(tabulate(nominal_result.sort_values(by=['U'],ascending=True),headers=li
st(nominal_result.columns.values),tablefmt = 'fancy_grid'))
print('NOMINAL (>2 GROUPS) (KRUSKAL-WALLIS):')
print(tabulate(nominal_3_result.sort_values(by=['H'],ascending=False),headers
=list(nominal_3_result.columns.values),tablefmt = 'fancy_grid'))
```

```
#identified genetics to be highly correlated, decided to investigate
genes

def std_dev(data):
    mean = np.mean(data)
    std = np.sqrt(np.sum((data - mean)**2) / (len(data) - 1))
```

```
    return std

def mean(data):
    return np.mean(data)

df = pd.read_csv('GSE9624_series_matrix.txt', sep='\t', skiprows=75,
index_col='ID_REF', skipfooter=1, engine='python')

healthy = df.iloc[:, 1:7]
obese = df.iloc[:, 7:12]
healthy_std = healthy.apply(std_dev, axis=1)
obese_std = obese.apply(std_dev, axis=1)
healthy_mean = healthy.apply(mean, axis=1)
obese_mean = obese.apply(mean, axis=1)

from scipy.stats import ttest_ind
pvalues = []
for i in range(len(healthy.index)):
  t, p = ttest_ind(healthy.iloc[i,].tolist(), obese.iloc[i,].tolist())
  pvalues.append(p)

updown = []
for i in range(len(healthy.index)):
  if healthy_mean[i] > obese_mean[i]:
    updown.append("Down")
  elif healthy_mean[i] < obese_mean[i]:
    updown.append("Up")
  else:
    updown.append("-")

result = pd.DataFrame({'ID_REF': df.index, 'healthy_mean': healthy_mean,
'healthy_std_dev': healthy_std, 'obese_mean': obese_mean, 'obese_std_dev':
obese_std , 't_test_p_value': pvalues,
'up_or_downregulated_in_obese':updown})
print(result)
```

```
#narrowing down the dataset of 54675 genes to 531
results_considered =
result[(result['healthy_std_dev']<0.5)&(result['obese_std_dev']<0.5)&(result[
't_test_p_value']<0.05)]
print(results_considered)
```

```
results_considered.to_csv('GSE9624_std_mean_t_test.tsv', sep='\t',
index=False)

#sorting and picking the top 50 genes with the smallest p-values (most
significant differences)

final = results_considered.sort_values(by=['t_test_p_value'])[:50]
```