In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans #安裝 conda install scikit-learn
```

In [2]:
```
#使用pandas套件的read_csv讀入資料,並以DataFram儲存(2D)
gda = pd.read_csv('diabetes.csv')
```

In [3]:
```
#檢查gda的資料型態(DataFrame是二維的結構)
type(gda)
```

Out[3]:
```
pandas.core.frame.DataFrame
```

In [4]:
```
#印出gda的內容
#顯示的結果在最前面多一個索引的欄位(pandas預設的),流水號, 從0開始
gda
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

In [5]:
```
#印出gda的資訊, 查看各個欄位的基本資料:名稱,非空值數, 資料型態。
#欄位同樣有預設的索引(pandas預設的),流水號, 從0開始
gda.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [6]:
```
#另一種選擇欄位的方法是使用欄位名稱
df = gda[['Glucose', 'DiabetesPedigreeFunction', 'Age']]
df.head()
```

Out[6]:

	Glucose	DiabetesPedigreeFunction	Age
0	148	0.627	50
1	85	0.351	31
2	183	0.672	32
3	89	0.167	21
4	137	2.288	33

In [7]:
```
#查看裁切出的data各個欄位的統計值
df.describe()
```

Out[7]:

	Glucose	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000
mean	120.894531	0.471876	33.240885
std	31.972618	0.331329	11.760232
min	0.000000	0.078000	21.000000
25%	99.000000	0.243750	24.000000
50%	117.000000	0.372500	29.000000
75%	140.250000	0.626250	41.000000
max	199.000000	2.420000	81.000000

In [8]:
```
#最小值最大值正規化(Min-Max Normalization)
from sklearn.preprocessing import MinMaxScaler
#建立轉換資料的框架 scaler, 範圍在 0~1 之間
scaler = MinMaxScaler(feature_range=(0,1))
```

In [9]:
```
#正式做資料轉換,其中先將資料(dataFrame)轉換為 numpy,再用 scale框架轉換資料
df_minmax = scaler.fit_transform(df.to_numpy())
#檢視轉換後的內容(介於0~1之間), 此時的資料型態是 numpy(2d)
df_minmax
```

Out[9]:
```
array([[0.74371859, 0.23441503, 0.48333333],
       [0.42713568, 0.11656704, 0.16666667],
       [0.91959799, 0.25362938, 0.18333333],
       ...,
       [0.6080402 , 0.07130658, 0.15      ],
       [0.63316583, 0.11571307, 0.43333333],
       [0.46733668, 0.10119556, 0.03333333]])
```

In [10]:
```
#後面集群分析時需要, 因此再轉回pandas的 DataFrame, 同時指定欄位名稱'Glucose', 'D
#轉換完後顯示資料, 會發現主動給予列索引
df_minmax = pd.DataFrame(df_minmax, columns = ['Glucose', 'DiabetesPedigreeF
df_minmax
```

Out[10]:

	Glucose	DiabetesPedigreeFunction	Age
0	0.743719	0.234415	0.483333
1	0.427136	0.116567	0.166667
2	0.919598	0.253629	0.183333
3	0.447236	0.038002	0.000000
4	0.688442	0.943638	0.200000
...	...	...	...
763	0.507538	0.039710	0.700000
764	0.613065	0.111870	0.100000
765	0.608040	0.071307	0.150000
766	0.633166	0.115713	0.433333
767	0.467337	0.101196	0.033333

768 rows × 3 columns

In [11]:
```
#使用前面的 from sklearn.cluster import KMeans 套件的集群分析函數 KMeans()
#先建立模型的框架。
#KMeans有很多參數可以設定, 不一定每個參數都需要, 若未設定則使自動使用預設值
# n_clusters=4 分群的數量,預設8群,在此設定4群。
# max_iter=500 分群過程中演算法執行最大迭代數, 在k-means中，如果執行結果收斂的話
#random_state 指定隨機亂數種子, 確保每次分群結果都一樣(若是要調參數,才能比較調整
model_KMC = KMeans(n_clusters = 4, max_iter = 500, random_state = 42)
```

In [12]:
```
#將資料進行分群(計算相似度高(距離較近)的點會被分為同一群)
#分群完後, 每一筆資料會給予一個群編號標籤,0, 1, 2, 3 =>4群
model_KMC.fit_predict(df_minmax)
```

```
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

Out[12]:
```
array([0, 1, 2, 1, 3, 1, 1, 1, 0, 0, 1, 2, 3, 0, 0, 1, 1, 1, 1, 1, 3, 0,
       2, 1, 0, 0, 0, 1, 0, 1, 0, 2, 1, 1, 0, 3, 2, 0, 1, 3, 2, 3, 0, 0,
       2, 3, 2, 1, 1, 1, 1, 1, 1, 0, 0, 1, 2, 3, 3, 1, 1, 2, 1, 2, 0, 1,
       3, 0, 1, 2, 3, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 2, 2, 1, 1, 1, 1, 3, 2, 1, 1, 3, 3, 1, 2, 1, 1,
       2, 0, 1, 1, 2, 0, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 1, 1, 0, 0, 2, 3,
       2, 1, 1, 1, 1, 1, 3, 1, 0, 1, 1, 0, 2, 1, 1, 3, 0, 1, 2, 1, 3, 2,
       2, 2, 1, 3, 1, 0, 2, 0, 1, 1, 3, 3, 2, 3, 1, 1, 1, 2, 1, 1, 1, 2,
       1, 1, 0, 3, 1, 1, 1, 1, 2, 2, 0, 3, 1, 2, 1, 0, 2, 0, 1, 2, 1, 1,
       3, 2, 3, 2, 3, 1, 0, 1, 0, 0, 1, 2, 1, 2, 0, 2, 1, 2, 1, 1, 3, 0,
       2, 0, 1, 0, 1, 1, 1, 2, 3, 1, 2, 0, 1, 1, 1, 2, 0, 2, 2, 1, 1, 1,
       2, 3, 2, 3, 0, 2, 1, 1, 0, 1, 1, 1, 3, 1, 1, 1, 2, 3, 2, 2, 1, 0,
       1, 3, 3, 3, 1, 2, 3, 1, 0, 1, 0, 1, 1, 1, 0, 3, 2, 0, 2, 0, 0, 0,
       2, 3, 1, 1, 1, 1, 3, 1, 0, 2, 2, 1, 0, 0, 2, 2, 1, 1, 2, 1, 0, 1,
       3, 3, 1, 1, 2, 1, 3, 1, 1, 2, 1, 0, 2, 1, 1, 0, 1, 2, 3, 2, 1, 1,
       3, 1, 2, 0, 1, 2, 3, 0, 3, 2, 1, 1, 1, 1, 0, 0, 2, 1, 1, 1, 1, 2,
       0, 1, 1, 0, 3, 0, 0, 2, 2, 0, 0, 0, 2, 1, 1, 1, 1, 0, 3, 3, 1, 1,
       3, 0, 1, 1, 2, 3, 3, 1, 3, 3, 1, 1, 3, 0, 0, 3, 0, 2, 1, 1, 2, 3,
       3, 1, 1, 2, 1, 0, 2, 1, 2, 1, 0, 1, 2, 2, 1, 1, 3, 2, 2, 2, 3, 2,
       1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 0, 1, 1, 1, 2, 3, 2, 0, 2, 1, 3,
       2, 1, 1, 3, 1, 3, 1, 1, 1, 1, 1, 2, 1, 0, 1, 2, 0, 1, 0, 0, 0, 1,
       1, 1, 3, 1, 1, 1, 1, 2, 2, 1, 1, 0, 1, 0, 1, 1, 0, 0, 2, 1, 1, 1,
       2, 2, 2, 0, 1, 0, 1, 0, 1, 3, 1, 0, 1, 1, 0, 2, 1, 1, 1, 3, 1, 1,
       2, 1, 3, 0, 0, 2, 0, 1, 1, 2, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 1, 3, 1, 0, 0, 3, 3, 0, 1, 0, 1, 1, 2, 0, 1, 0, 2,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 2, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1,
       1, 1, 3, 1, 3, 1, 2, 0, 2, 1, 0, 0, 0, 1, 0, 1, 0, 1, 3, 1, 0, 3,
       3, 2, 0, 1, 2, 1, 1, 1, 1, 0, 2, 1, 3, 1, 2, 1, 1, 2, 2, 3, 0, 1,
       1, 1, 3, 1, 1, 3, 3, 1, 1, 1, 1, 1, 0, 1, 3, 1, 1, 1, 1, 1, 0, 1,
       3, 1, 1, 1, 0, 1, 1, 2, 2, 2, 0, 1, 1, 1, 1, 1, 1, 2, 1, 3, 0, 3,
       0, 3, 2, 2, 0, 1, 0, 0, 0, 0, 0, 1, 0, 3, 0, 2, 0, 1, 1, 1, 1, 2,
       1, 1, 0, 1, 1, 1, 3, 0, 3, 0, 3, 0, 1, 0, 2, 1, 1, 3, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 1, 2, 0, 2, 1, 1, 2, 2, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       1, 2, 2, 1, 2, 1, 2, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 3, 0, 2, 3,
       2, 0, 3, 1, 1, 2, 0, 3, 2, 0, 1, 0, 1, 2, 1, 0, 1, 1, 0, 1])
```

In [13]:
```
#Series是Pandas的一維陣列結構。
#將model內的標籤存入 Series,再以不同的值分別計數, 印出每一群的個數
r1 = pd.Series(model_KMC.labels_).value_counts()
r1
```

Out[13]:
```
1    362
0    165
2    143
3     98
dtype: int64
```

In [14]:
```
#將model內的中心點(座標)印出
r2 = pd.DataFrame(model_KMC.cluster_centers_)
r2
```

Out[14]:

	0	1	2
0	0.659845	0.139052	0.503939
1	0.507121	0.115221	0.089733
2	0.802193	0.152407	0.160373
3	0.606143	0.435856	0.184864

In [15]:
```
#合併上述兩個矩陣 (axis=1表示欄合併),得到集群中心和筆數的矩陣
r21 = pd.concat([r2, r1], axis = 1)
r21
```

Out[15]:

	0	1	2	0
0	0.659845	0.139052	0.503939	165
1	0.507121	0.115221	0.089733	362
2	0.802193	0.152407	0.160373	143
3	0.606143	0.435856	0.184864	98

In [16]:
```
#重新命名 欄位名稱
# list+list
r21.columns = list(df_minmax.columns) + ['size_of_Group']
r21
```

Out[16]:

	Glucose	DiabetesPedigreeFunction	Age	size_of_Group
0	0.659845	0.139052	0.503939	165
1	0.507121	0.115221	0.089733	362
2	0.802193	0.152407	0.160373	143
3	0.606143	0.435856	0.184864	98

In [17]:
```
#將標準化資料 df_minmax 增加群別的欄位 model_KMC.labels_ (axis=1表示欄合併)
r = pd.concat([df_minmax, pd.Series(model_KMC.labels_, index = df_minmax.inc
r
```

Out[17]:

	Glucose	DiabetesPedigreeFunction	Age	0
0	0.743719	0.234415	0.483333	0
1	0.427136	0.116567	0.166667	1
2	0.919598	0.253629	0.183333	2
3	0.447236	0.038002	0.000000	1
4	0.688442	0.943638	0.200000	3
...	...	...	...	...
763	0.507538	0.039710	0.700000	0
764	0.613065	0.111870	0.100000	1
765	0.608040	0.071307	0.150000	1
766	0.633166	0.115713	0.433333	0
767	0.467337	0.101196	0.033333	1

768 rows × 4 columns

In [18]:
```
#重新命名 欄位名稱
r.columns = list(df_minmax.columns) + ['GroupID']
r
```

Out[18]:

	Glucose	DiabetesPedigreeFunction	Age	GroupID
0	0.743719	0.234415	0.483333	0
1	0.427136	0.116567	0.166667	1
2	0.919598	0.253629	0.183333	2
3	0.447236	0.038002	0.000000	1
4	0.688442	0.943638	0.200000	3
...	...	...	...	...
763	0.507538	0.039710	0.700000	0
764	0.613065	0.111870	0.100000	1
765	0.608040	0.071307	0.150000	1
766	0.633166	0.115713	0.433333	0
767	0.467337	0.101196	0.033333	1

768 rows × 4 columns

In [19]:
```
#原始資料 gda 增加一個欄位 "k-means", 其值為分群後的群別
gda["k-means"] = model_KMC.labels_
gda
```

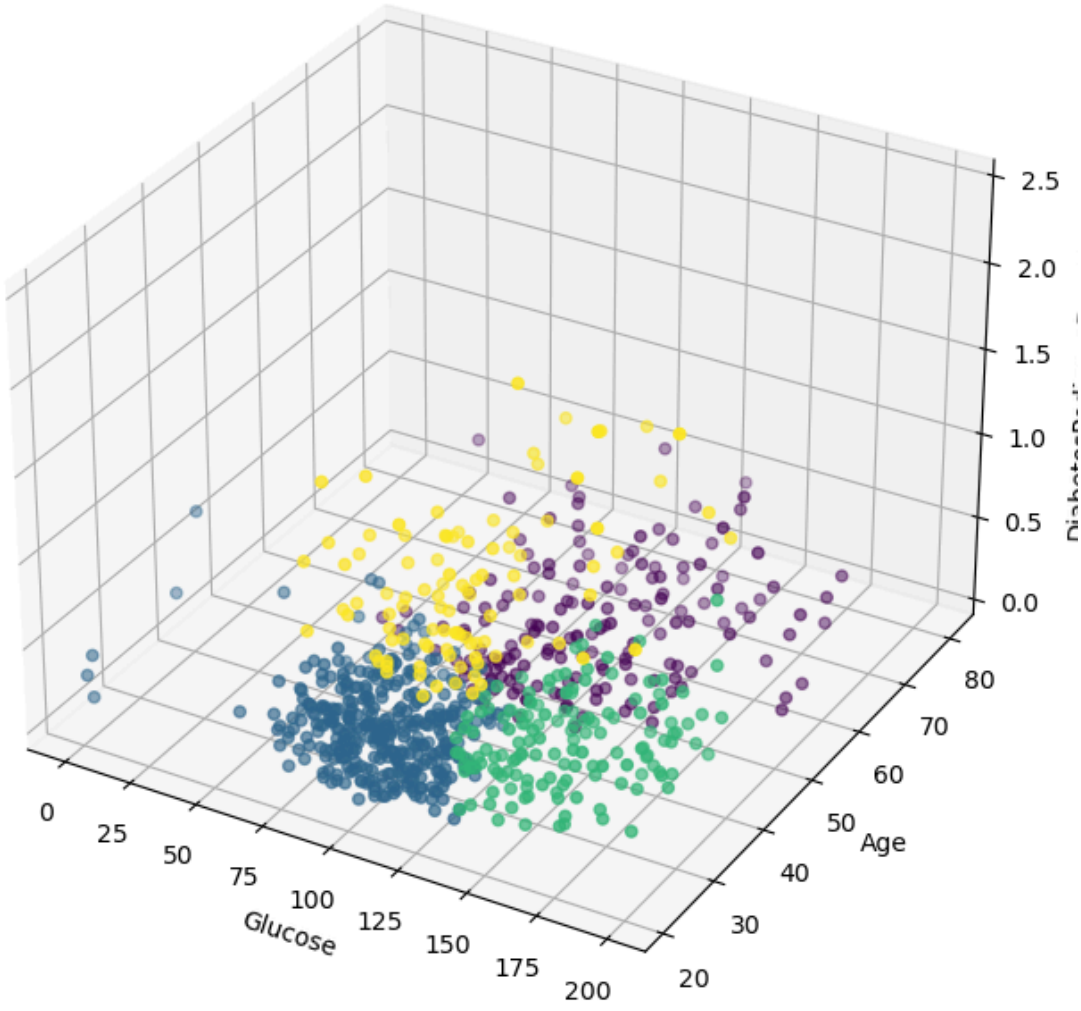
Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 10 columns

In [20]:
```
#將上述結果儲存為 csv 檔
gda.to_csv("gda_KMC_Output.csv",index = False)
```

In [21]:
```
#3D散佈圖_精簡方法
import matplotlib.pyplot as plt
dft = pd.read_csv('gda_KMC_Output.csv') #讀取csv資料
#設定 10*8 大小的空畫布
fig = plt.figure(figsize = (10, 8))
#設定 3D 圖
ax = fig.add_subplot(111, projection='3d')
#繪製散佈圖, 給予三軸資料,
# c(color)使用預設的顏色編號,可以指定 cmap='',不同顏色表示不同群
# marker = 'o' 標記為圖形
ax.scatter(dft['Glucose'], dft['Age'], dft['DiabetesPedigreeFunction'], c =
ax.set_xlabel('Glucose')
ax.set_ylabel('Age')
ax.set_zlabel('DiabetesPedigreeFunction')
plt.show()
```



In [22]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

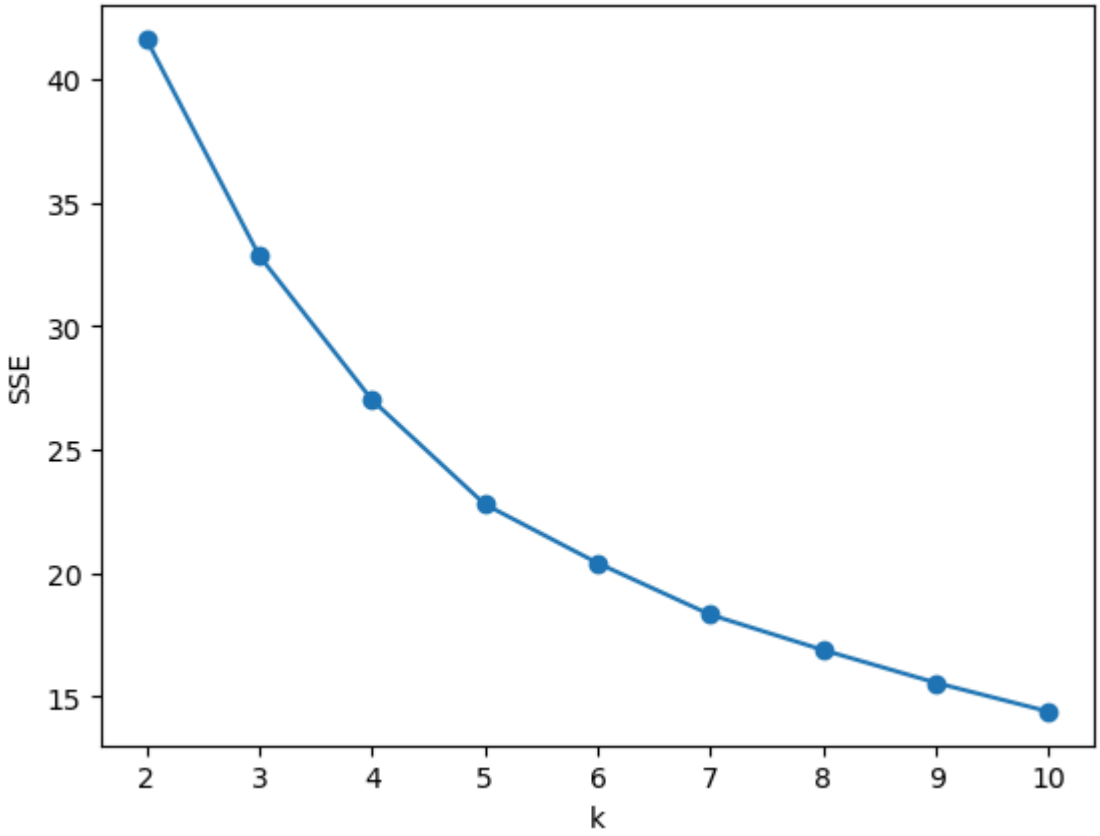
In [23]:
```
#設定不同的分群數, 觀察SSE值, 找出適合的分群數
#模型.inertia_ 記錄分群後的 SSE
SSE = []
for k in range(2, 11):
 est = KMeans(n_clusters = k)
 est.fit(df_minmax)
 SSE.append(est.inertia_)
SSE
```

```
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
```

```
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

Out[23]:
```
[41.63343773897377,
 32.85396576606729,
 27.00455790785569,
 22.775189393004553,
 20.391810143335668,
 18.29855455488949,
 16.86639623919868,
 15.537684340308854,
 14.36872254462624]
```

In [24]:
```
# 繪製散佈圖, 找到 Elbow(手肘)點
# 從圖中可以看出 elbow 出現在 4
X = range(2, 11)
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X, SSE, 'o-')
plt.show()
```



In [25]:
```
#將每一群客戶資料抽出另外儲存
group0 = r[r['GroupID'] == 0]
group1 = r[r['GroupID'] == 1]
group2 = r[r['GroupID'] == 2]
group3 = r[r['GroupID'] == 3]
```

In [26]:
```
#每一群的基本統計量描述
group0.describe()
```

Out[26]:

	Glucose	DiabetesPedigreeFunction	Age	GroupID
count	165.000000	165.000000	165.000000	165.0
mean	0.659845	0.139052	0.503939	0.0
std	0.145246	0.098973	0.142779	0.0
min	0.286432	0.002989	0.300000	0.0
25%	0.547739	0.066183	0.400000	0.0
50%	0.648241	0.105038	0.483333	0.0
75%	0.743719	0.212212	0.616667	0.0
max	0.989950	0.461571	1.000000	0.0

In [27]:
```
group1.describe()
```

Out[27]:

	Glucose	DiabetesPedigreeFunction	Age	GroupID
count	362.000000	362.000000	362.000000	362.0
mean	0.507121	0.115221	0.089733	1.0
std	0.102635	0.073435	0.086960	0.0
min	0.000000	0.000000	0.000000	1.0
25%	0.452261	0.055081	0.016667	1.0
50%	0.512563	0.096285	0.066667	1.0
75%	0.577889	0.172822	0.133333	1.0
max	0.688442	0.296755	0.350000	1.0

In [28]:
```
group2.describe()
```

Out[28]:

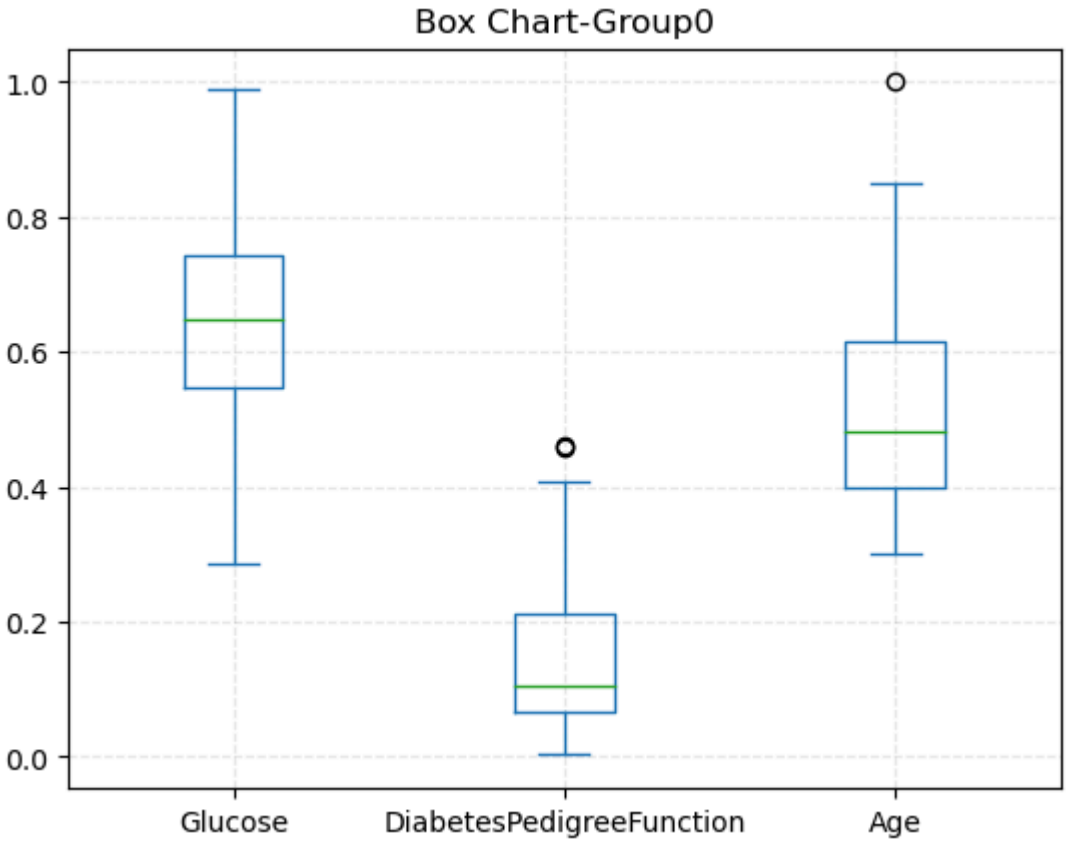
	Glucose	DiabetesPedigreeFunction	Age	GroupID
count	143.000000	143.000000	143.000000	143.0
mean	0.802193	0.152407	0.160373	2.0
std	0.094558	0.092000	0.106752	0.0
min	0.648241	0.004270	0.000000	2.0
25%	0.723618	0.080273	0.066667	2.0
50%	0.788945	0.126815	0.133333	2.0
75%	0.876884	0.216695	0.250000	2.0
max	0.994975	0.475235	0.366667	2.0

In [29]:
```
group3.describe()
```

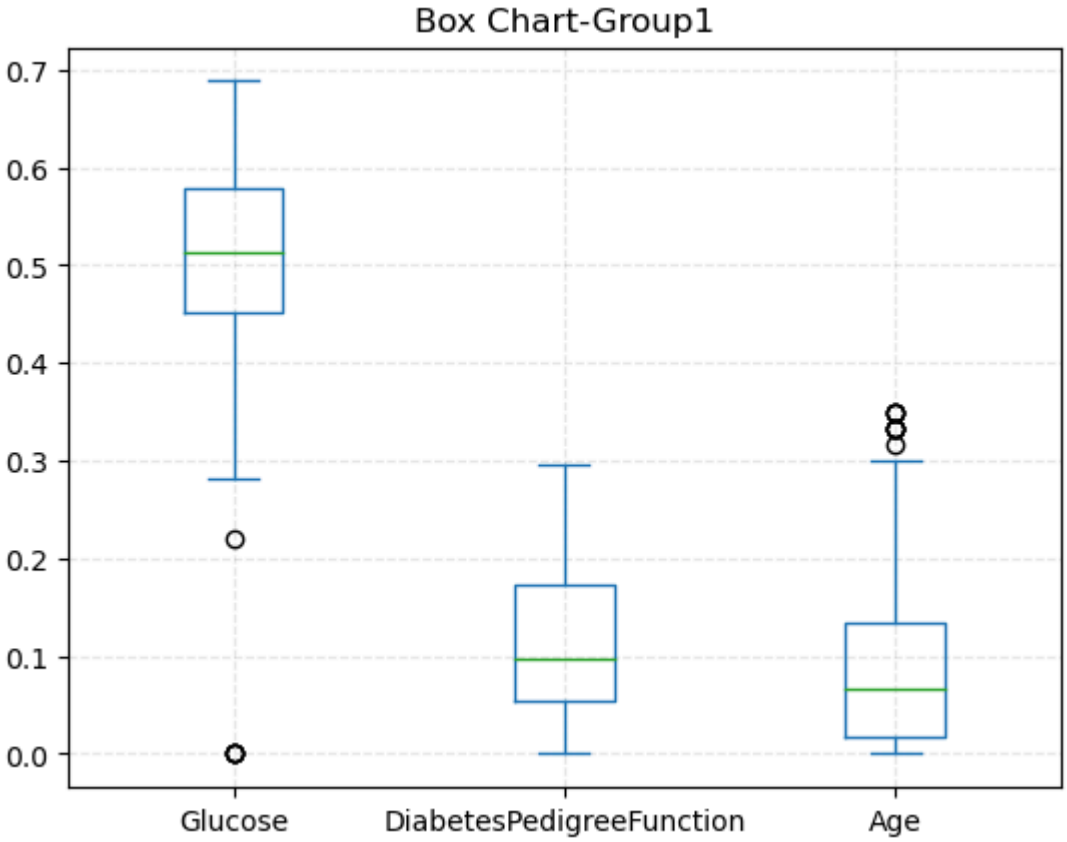
Out[29]:

	Glucose	DiabetesPedigreeFunction	Age	GroupID
count	98.000000	98.000000	98.000000	98.0
mean	0.606143	0.435856	0.184864	3.0
std	0.134191	0.156750	0.139814	0.0
min	0.386935	0.248506	0.000000	3.0
25%	0.523869	0.337319	0.066667	3.0
50%	0.592965	0.378309	0.183333	3.0
75%	0.648241	0.497972	0.262500	3.0
max	1.000000	1.000000	0.600000	3.0

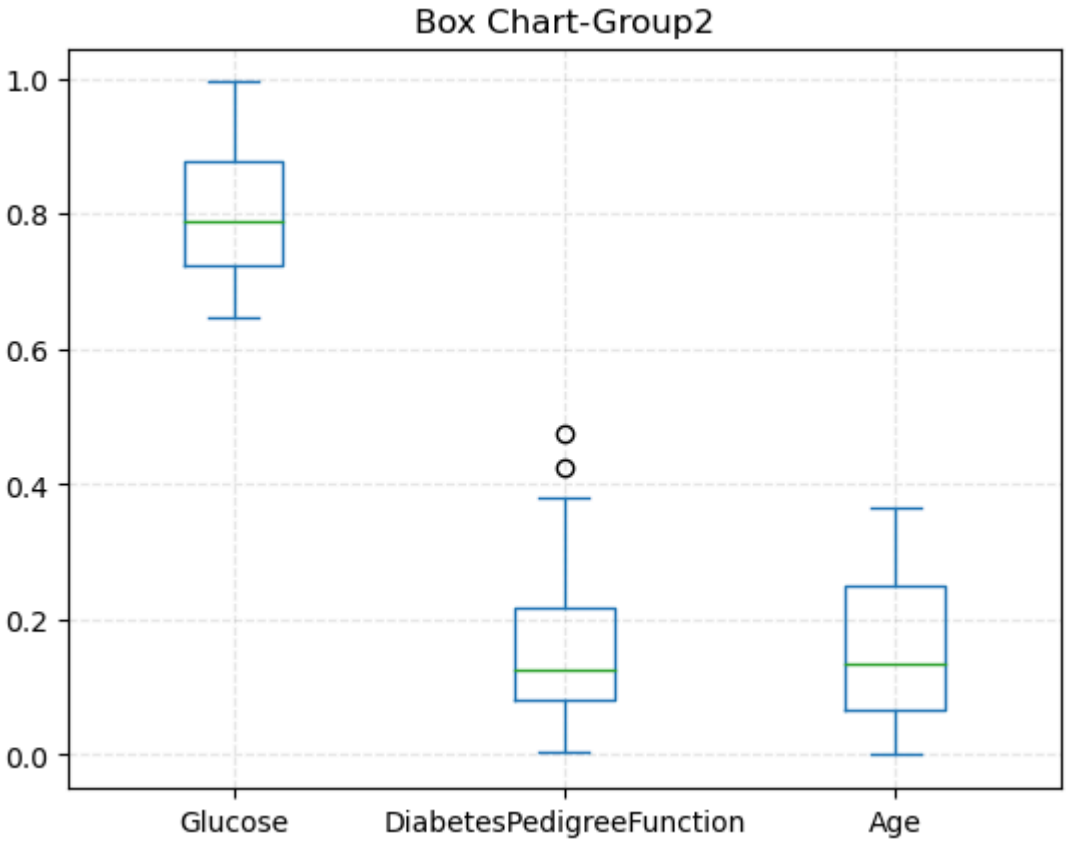
In [30]:
```
#繪製盒鬚圖(箱型圖)
#重點在觀察中位數的高低和離群值(outlier)的大小
#盒鬚圖的頂端至底端分別表示了: 最大值,上四分位數,中位數,下四分位數,最小值。超出圖形
group0 = group0[['Glucose', 'DiabetesPedigreeFunction', 'Age']]
group0.plot.box(title = "Box Chart-Group0")
plt.grid(linestyle = "--", alpha = 0.3)
```



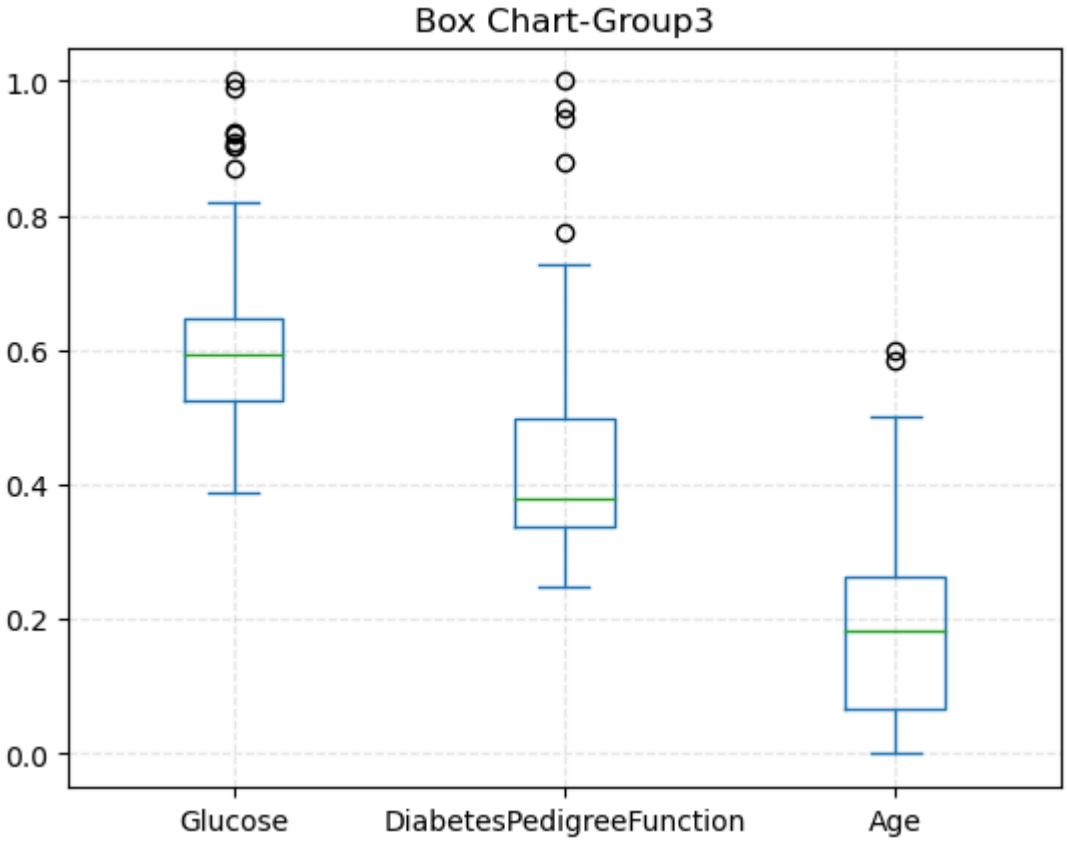
In [31]:
```
group1 = group1[['Glucose', 'DiabetesPedigreeFunction', 'Age']]
group1.plot.box(title = "Box Chart-Group1")
plt.grid(linestyle = "--", alpha = 0.3)
```



In [32]:
```
group2 = group2[['Glucose', 'DiabetesPedigreeFunction', 'Age']]
group2.plot.box(title = "Box Chart-Group2")
plt.grid(linestyle = "--", alpha = 0.3)
```

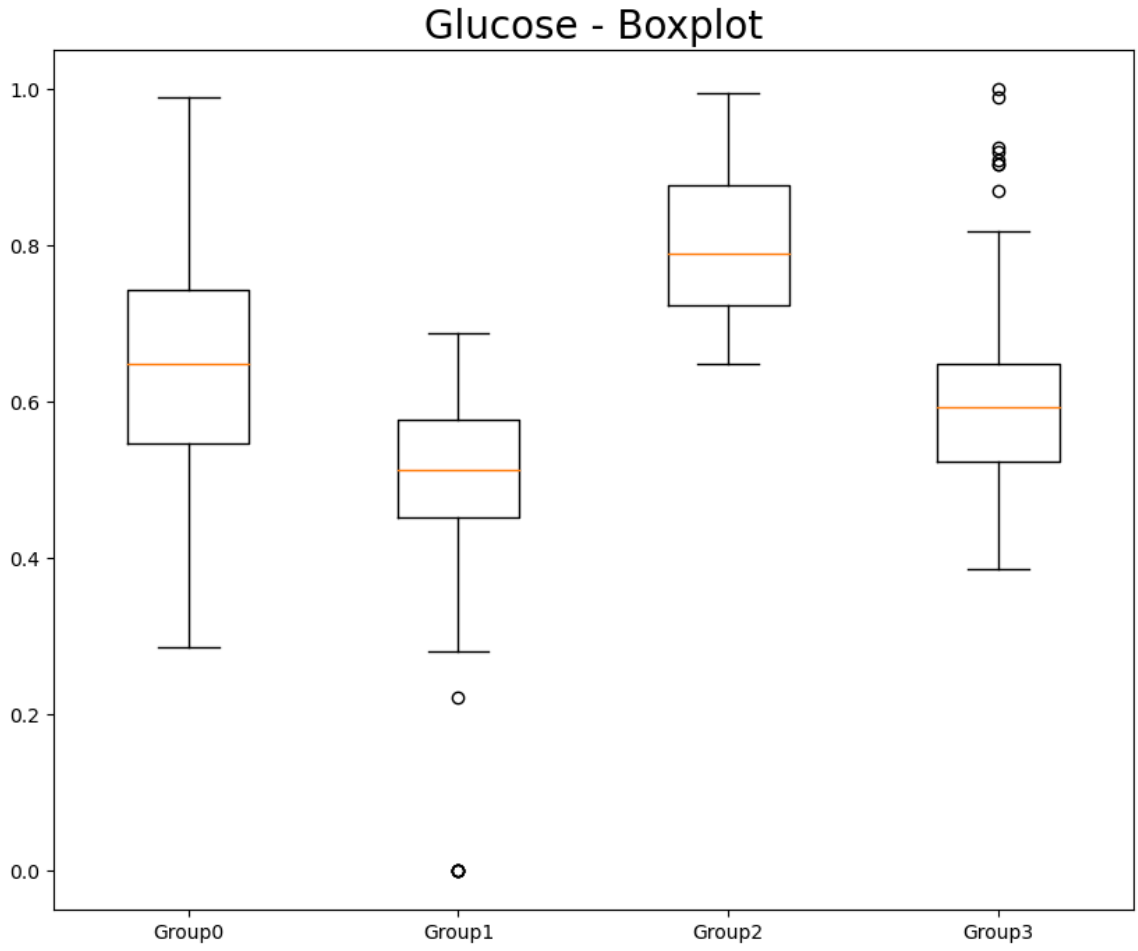


In [33]:
```
group3 = group3[['Glucose', 'DiabetesPedigreeFunction', 'Age']]
group3.plot.box(title = "Box Chart-Group3")
plt.grid(linestyle = "--", alpha = 0.3)
```

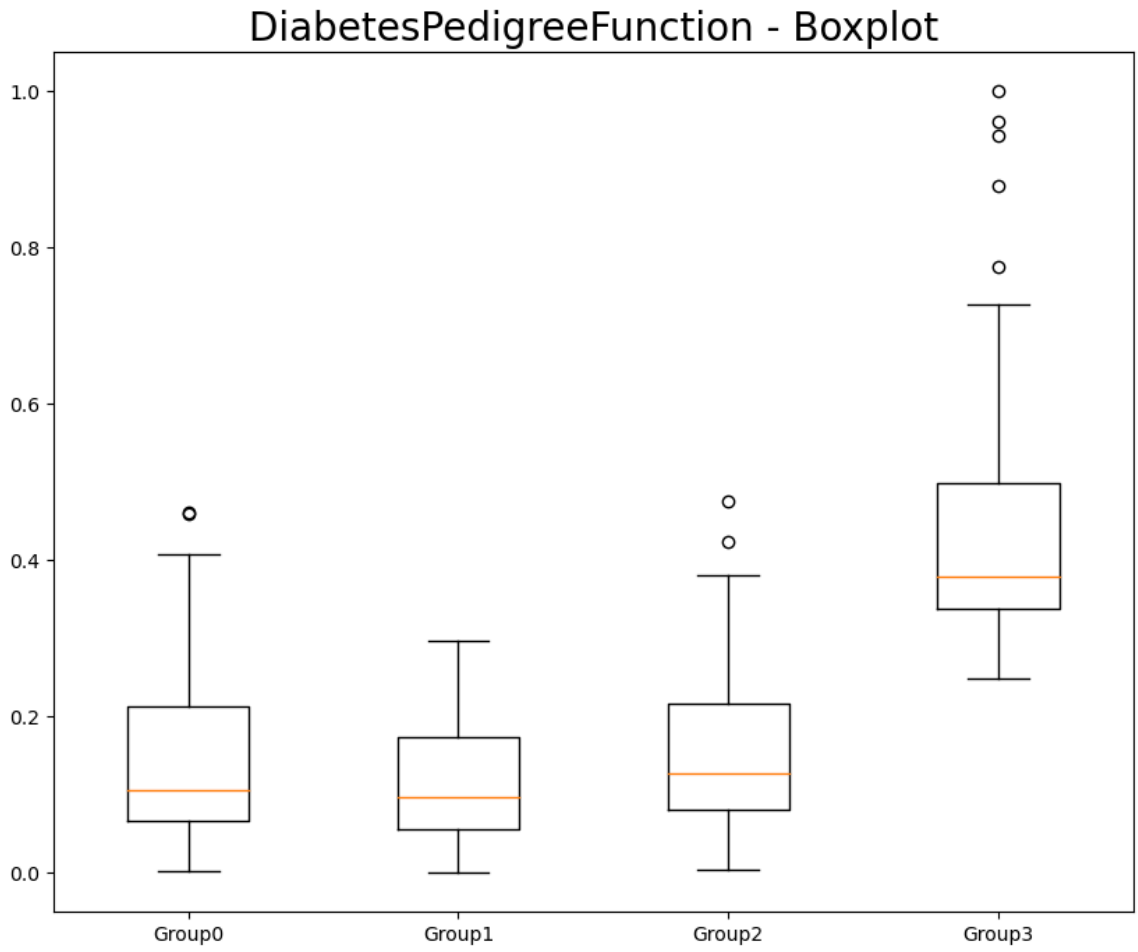


In [34]:
```
#重新編排資料, 改為一次看四個群的同一個欄位值
g0 = r[r['GroupID'] == 0]
g1 = r[r['GroupID'] == 1]
g2 = r[r['GroupID'] == 2]
g3 = r[r['GroupID'] == 3]
```

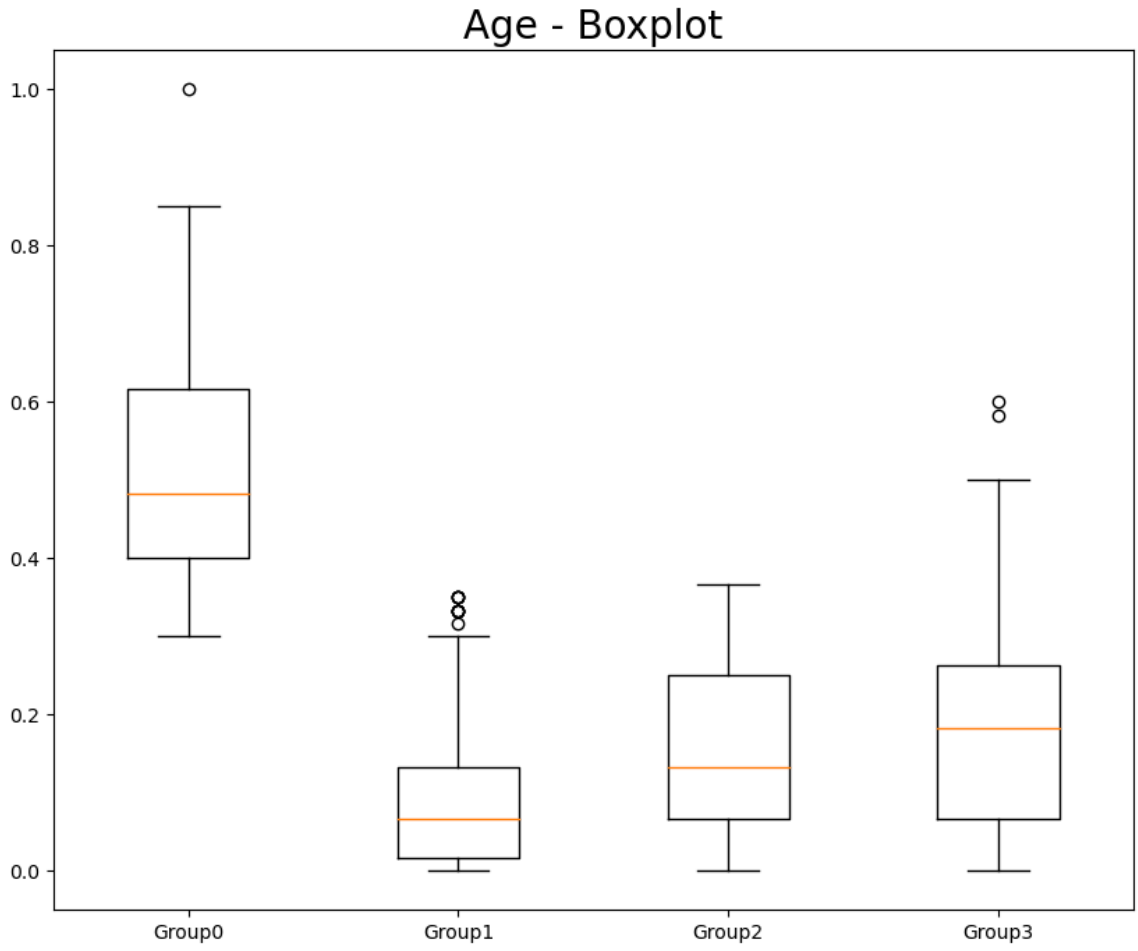
In [35]:
```
# Glucose
plt.figure(figsize = (10, 8))
labels = 'Group0', 'Group1', 'Group2', 'Group3'
plt.boxplot([g0['Glucose'], g1['Glucose'], g2['Glucose'], g3['Glucose']], la
plt.title('Glucose - Boxplot', fontsize = 20)
plt.show()
```



In [36]:
```
# DiabetesPedigreeFunction
plt.figure(figsize = (10, 8))
labels = 'Group0', 'Group1', 'Group2', 'Group3'
plt.boxplot([g0['DiabetesPedigreeFunction'], g1['DiabetesPedigreeFunction'],
plt.title('DiabetesPedigreeFunction - Boxplot', fontsize = 20)
plt.show()
```



In [37]:
```
# Age
plt.figure(figsize = (10, 8))
labels = 'Group0', 'Group1', 'Group2', 'Group3'
plt.boxplot([g0['Age'], g1['Age'], g2['Age'], g3['Age']], labels = labels)
plt.title('Age - Boxplot', fontsize = 20)
plt.show()
```



In [ ]:
```
```