In [55]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statistics as st
import warnings
import os
warnings.filterwarnings("ignore")
sns.set(rc={"figure.figsize":(15,6)})
pd.pandas.set_option("display.max_columns",None)
%matplotlib inline
```

In [56]:

```
root = "/content/drive/MyDrive/Colab_Notebooks"
os.chdir(root)
```

In [57]:

```
data = pd.read_csv("CVD_cleaned.csv")
```

In [58]:

```
data.head()
```

Out[58]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depressio
0	Poor	Within the past 2 years	No	No	No	No	N
1	Very Good	Within the past year	No	Yes	No	No	N
2	Very Good	Within the past year	Yes	No	No	No	N
3	Poor	Within the past year	Yes	Yes	No	No	N
4	Good	Within the past year	No	No	No	No	N

In [59]:

```
data.tail()
```

Out[59]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Dep
308849	Very Good	Within the past year	Yes	No	No	No	
308850	Fair	Within the past 5 years	Yes	No	No	No	
308851	Very Good	5 or more years ago	Yes	No	No	No	
308852	Very Good	Within the past year	Yes	No	No	No	
308853	Excellent	Within the past year	Yes	No	No	No	

In [60]:

```
## Get The shape of data
data.shape
```

Out[60]:

```
(308854, 19)
```

In [61]:

```
# get information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   General_Health                308854 non-null  object 
 1   Checkup                       308854 non-null  object 
 2   Exercise                      308854 non-null  object 
 3   Heart_Disease                 308854 non-null  object 
 4   Skin_Cancer                   308854 non-null  object 
 5   Other_Cancer                  308854 non-null  object 
 6   Depression                    308854 non-null  object 
 7   Diabetes                      308854 non-null  object 
 8   Arthritis                     308854 non-null  object 
 9   Sex                           308854 non-null  object 
 10  Age_Category                  308854 non-null  object 
 11  Height_(cm)                   308854 non-null  float64
 12  Weight_(kg)                   308854 non-null  float64
 13  BMI                           308854 non-null  float64
 14  Smoking_History               308854 non-null  object 
 15  Alcohol_Consumption           308854 non-null  float64
 16  Fruit_Consumption             308854 non-null  float64
 17  Green_Vegetables_Consumption  308854 non-null  float64
 18  FriedPotato_Consumption       308854 non-null  float64
dtypes: float64(7), object(12)
memory usage: 44.8+ MB
```

In [62]:

```
data.columns
```

Out[62]:

```
Index(['General_Health', 'Checkup', 'Exercise', 'Heart_Disease', 'Skin_Can
cer',
       'Other_Cancer', 'Depression', 'Diabetes', 'Arthritis', 'Sex',
       'Age_Category', 'Height_(cm)', 'Weight_(kg)', 'BMI', 'Smoking_Histo
ry',
       'Alcohol_Consumption', 'Fruit_Consumption',
       'Green_Vegetables_Consumption', 'FriedPotato_Consumption'],
      dtype='object')
```

In [63]:

```
# check with null values
data.isnull().sum()
```

Out[63]:

```
General_Health                  0
Checkup                         0
Exercise                        0
Heart_Disease                   0
Skin_Cancer                     0
Other_Cancer                    0
Depression                      0
Diabetes                        0
Arthritis                       0
Sex                             0
Age_Category                    0
Height_(cm)                     0
Weight_(kg)                     0
BMI                             0
Smoking_History                 0
Alcohol_Consumption             0
Fruit_Consumption               0
Green_Vegetables_Consumption    0
FriedPotato_Consumption         0
dtype: int64
```

In [64]:

```
# Check with duplicet values
data.duplicated().sum()
```

Out[64]:

```
80
```

In [72]:

```
# now we have to split numerical and catigorical columns
catigorical_features = data.select_dtypes(include="object").columns
numerical_features = data.select_dtypes(exclude="object").columns
print(catigorical_features)
print("="*100)
print(numerical_features)
```

```
Index(['General_Health', 'Checkup', 'Exercise', 'Heart_Disease', 'Skin_Can
cer',
       'Other_Cancer', 'Depression', 'Diabetes', 'Arthritis', 'Sex',
       'Smoking_History'],
      dtype='object')
====================================================================================
=========================
Index(['Age_Category', 'Height_(cm)', 'Weight_(kg)', 'BMI',
       'Alcohol_Consumption', 'Fruit_Consumption',
       'Green_Vegetables_Consumption', 'FriedPotato_Consumption'],
      dtype='object')
```

In [66]:

```
# now check the unique values in the catigorical data
try:
  for i in catigorical_features:
    print(i,data[i].unique())
    print("="*100)
except Exception as e:
  print(e)
```

```
General_Health ['Poor' 'Very Good' 'Good' 'Fair' 'Excellent']
====================================================================================
==========================
Checkup ['Within the past 2 years' 'Within the past year' '5 or more years
ago'
 'Within the past 5 years' 'Never']
====================================================================================
==========================
Exercise ['No' 'Yes']
====================================================================================
==========================
Heart_Disease ['No' 'Yes']
====================================================================================
==========================
Skin_Cancer ['No' 'Yes']
====================================================================================
==========================
Other_Cancer ['No' 'Yes']
====================================================================================
==========================
Depression ['No' 'Yes']
====================================================================================
==========================
Diabetes ['No' 'Yes' 'No, pre-diabetes or borderline diabetes'
 'Yes, but female told only during pregnancy']
====================================================================================
==========================
Arthritis ['Yes' 'No']
====================================================================================
==========================
Sex ['Female' 'Male']
====================================================================================
==========================
Age_Category ['70-74' '60-64' '75-79' '80+' '65-69' '50-54' '45-49' '18-2
4' '30-34'
 '55-59' '35-39' '40-44' '25-29']
====================================================================================
==========================
Smoking_History ['Yes' 'No']
====================================================================================
==========================
```

In [67]:

```
# check the value count of data
try:
  for i in catigorical_features:
    print(i,data[i].value_counts())
    print("="*100)
except Exception as e:
  print(e)
```

```
General_Health Very Good    110395
Good          95364
Excellent     55954
Fair          35810
Poor          11331
Name: General_Health, dtype: int64
================================================================================
==========================
Checkup Within the past year       239371
Within the past 2 years     37213
Within the past 5 years     17442
5 or more years ago         13421
Never                        1407
Name: Checkup, dtype: int64
================================================================================
==========================
Exercise Yes    239381
No      69473
Name: Exercise, dtype: int64
================================================================================
==========================
Heart_Disease No     283883
Yes     24971
Name: Heart_Disease, dtype: int64
================================================================================
==========================
Skin_Cancer No     278860
Yes     29994
Name: Skin_Cancer, dtype: int64
================================================================================
==========================
Other_Cancer No     278976
Yes     29878
Name: Other_Cancer, dtype: int64
================================================================================
==========================
Depression No     246953
Yes     61901
Name: Depression, dtype: int64
================================================================================
==========================
Diabetes No                                                 259141
Yes                                              40171
No, pre-diabetes or borderline diabetes           6896
Yes, but female told only during pregnancy        2646
Name: Diabetes, dtype: int64
================================================================================
==========================
Arthritis No     207783
Yes    101071
Name: Arthritis, dtype: int64
================================================================================
==========================
Sex Female    160196
Male      148658
Name: Sex, dtype: int64
================================================================================
==========================
Age_Category 65-69    33434
60-64    32418
70-74    31103
```

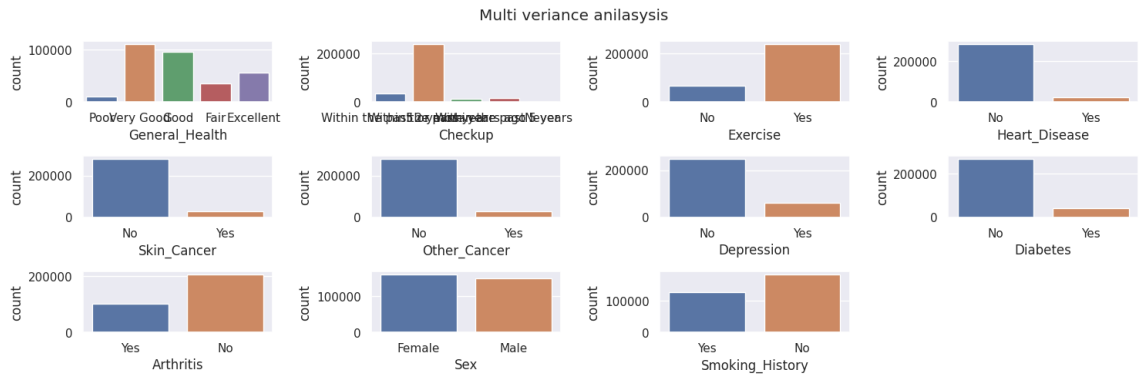
```
55-59    28054
50-54    25097
80+      22271
40-44    21595
45-49    20968
75-79    20705
35-39    20606
18-24    18681
30-34    18428
25-29    15494
Name: Age_Category, dtype: int64
================================================================================
==========================
Smoking_History No     183590
Yes    125264
Name: Smoking_History, dtype: int64
================================================================================
==========================
```

In [68]:

```
# Now WE have TO CLean The Data
data["Diabetes"] = data["Diabetes"].apply(lambda x:x.replace('No, pre-diabetes or border
data["Diabetes"] = data["Diabetes"].apply(lambda x:x.replace("Yes, but female told only
data["Age_Category"] = data["Age_Category"].apply(lambda x:"".join(x.split("-")[0])).str
```

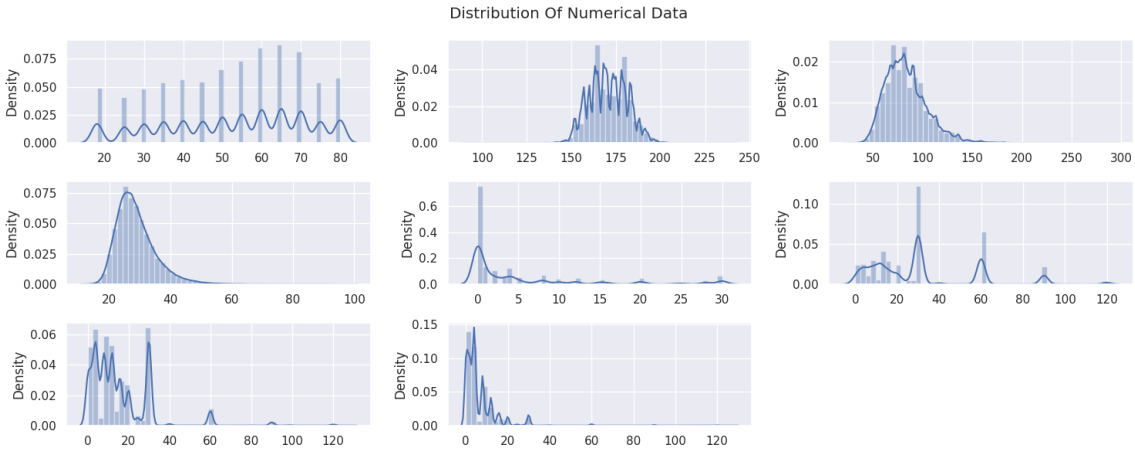
In [87]:

```
try:
  plt.suptitle("Multi veriance anilasysis")
  for i in range(len(catigorical_features)):
    plt.subplot(4,4,i+1)
    sns.countplot(x=data[catigorical_features[i]])
    plt.tight_layout()
except Exception as e:
  print(e)
```



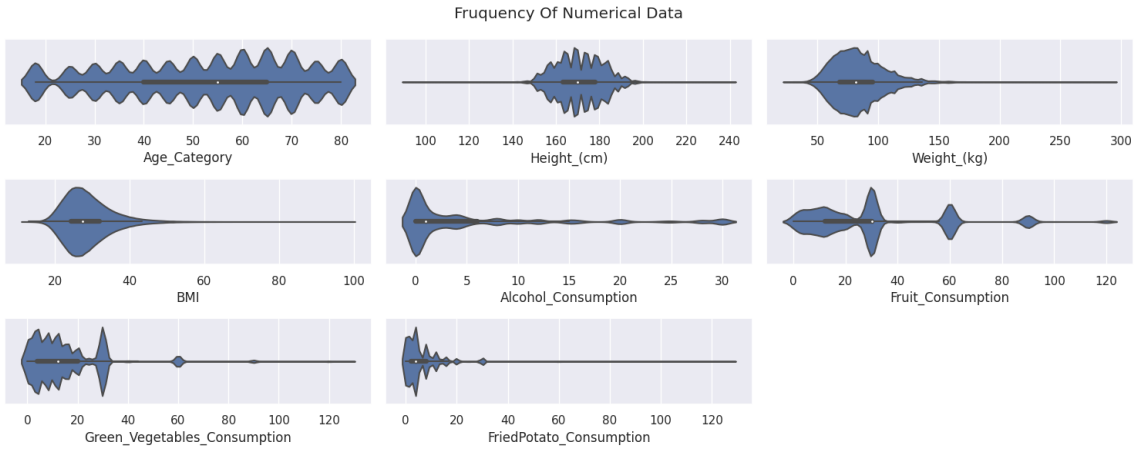
In [90]:

```
# now We Have To Check Distribution
try:
  plt.suptitle("Distribution Of Numerical Data")
  for i in range(len(numerical_features)):
    plt.subplot(3,3,i+1)
    sns.distplot(x=data[numerical_features[i]])
    plt.tight_layout()
except Exception as e:
  print(e)
```



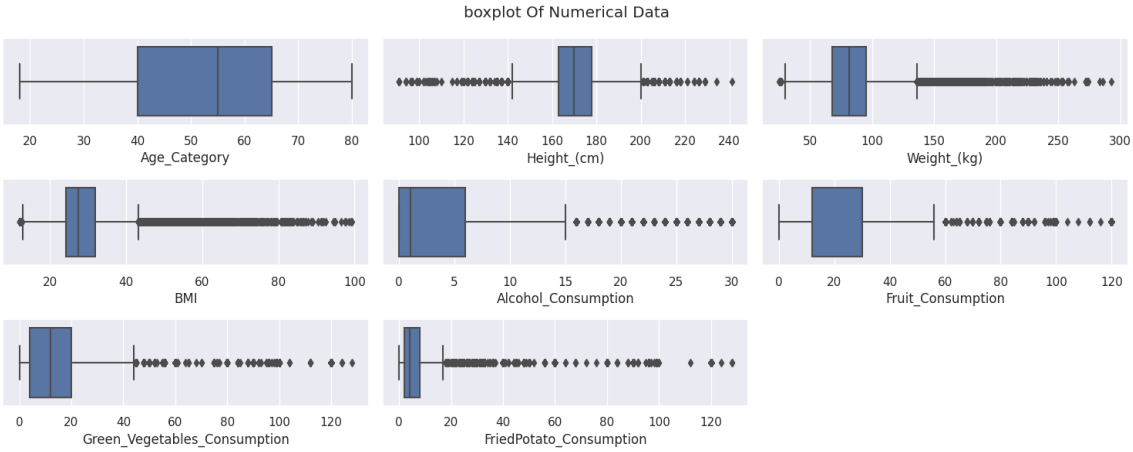
In [91]:

```
try:
  plt.suptitle("Fruquency Of Numerical Data")
  for i in range(len(numerical_features)):
    plt.subplot(3,3,i+1)
    sns.violinplot(x=data[numerical_features[i]])
    plt.tight_layout()
except Exception as e:
  print(e)
```



In [92]:

```
try:
  plt.suptitle("boxplot Of Numerical Data")
  for i in range(len(numerical_features)):
    plt.subplot(3,3,i+1)
    sns.boxplot(x=data[numerical_features[i]])
    plt.tight_layout()
except Exception as e:
  print(e)
```

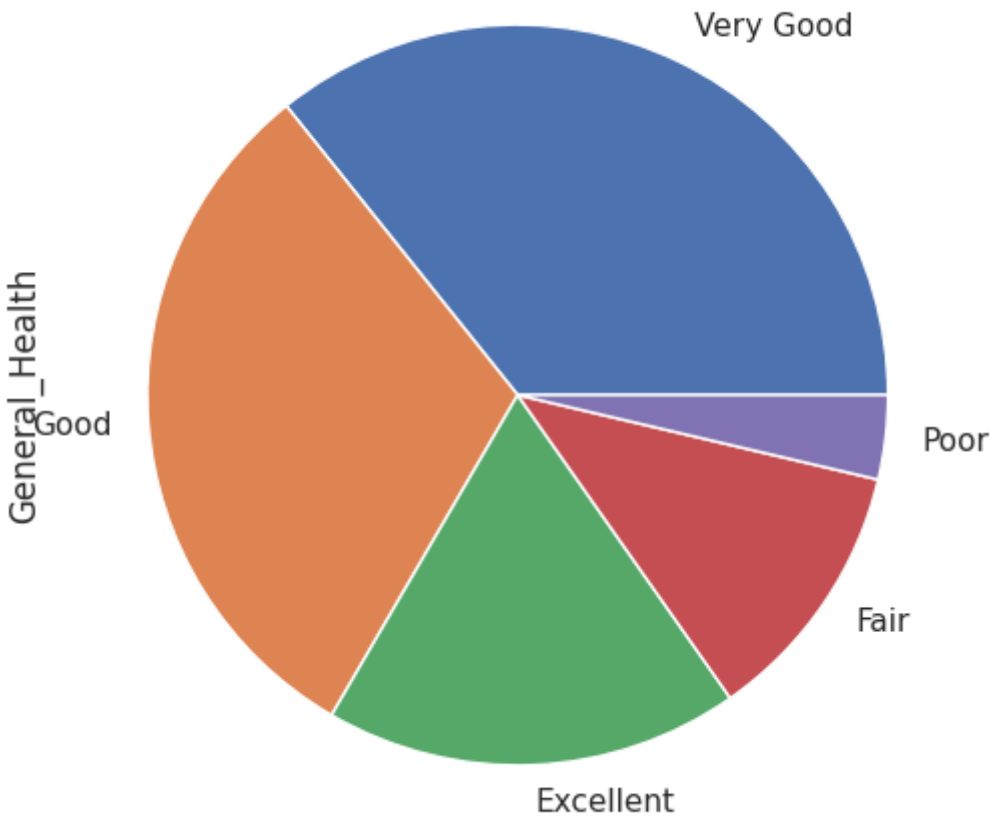


In [93]:

```
data["General_Health"].value_counts().plot.pie()
```

Out[93]:

```
<Axes: ylabel='General_Health'>
```

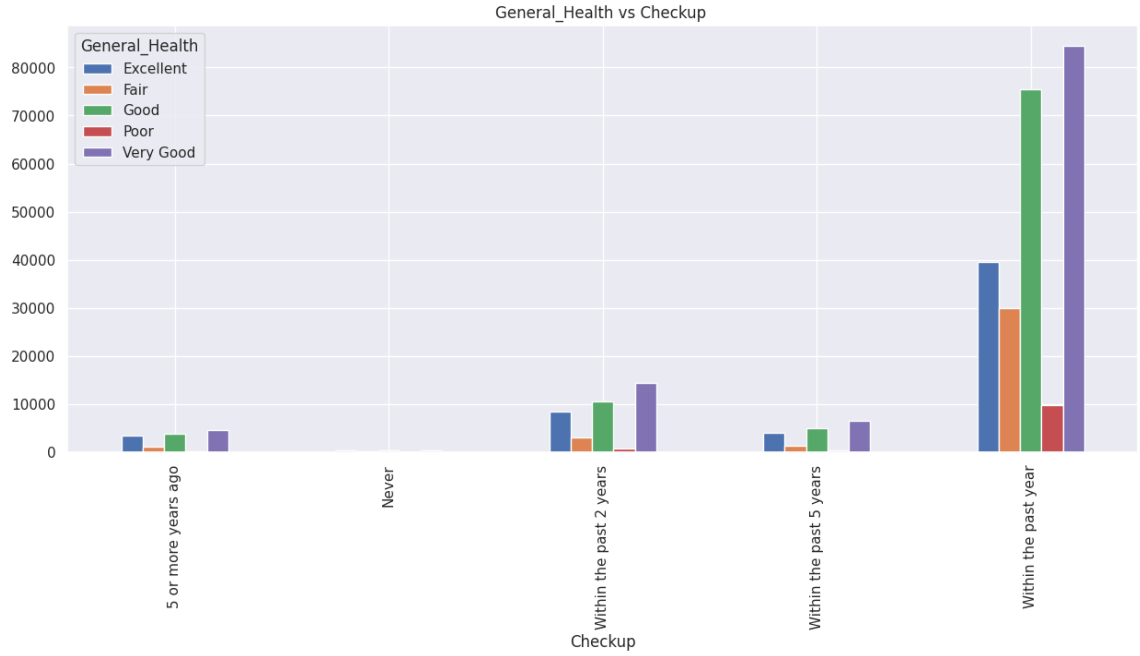


In [98]:

```
checkup = pd.crosstab(data["Checkup"],data["General_Health"])
checkup.plot(kind="bar")
plt.title("General_Health vs Checkup ")
```

Out[98]:

```
Text(0.5, 1.0, 'General_Health vs Checkup ')
```

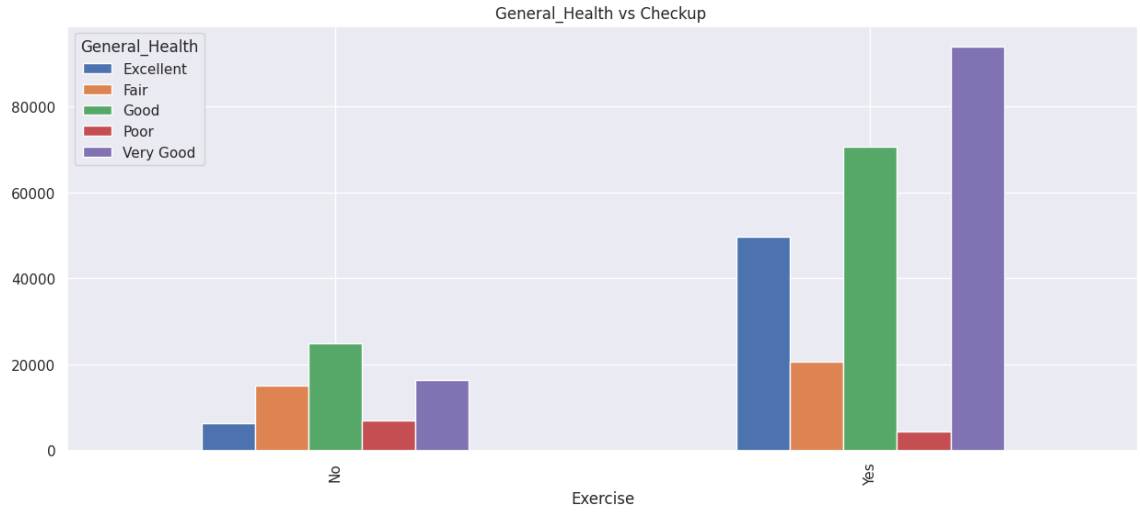


In [101]:

```
Exercise = pd.crosstab(data["Exercise"],data["General_Health"])
Exercise.plot(kind="bar")
plt.title("General_Health vs Checkup ")
```

Out[101]:

```
Text(0.5, 1.0, 'General_Health vs Checkup ')
```

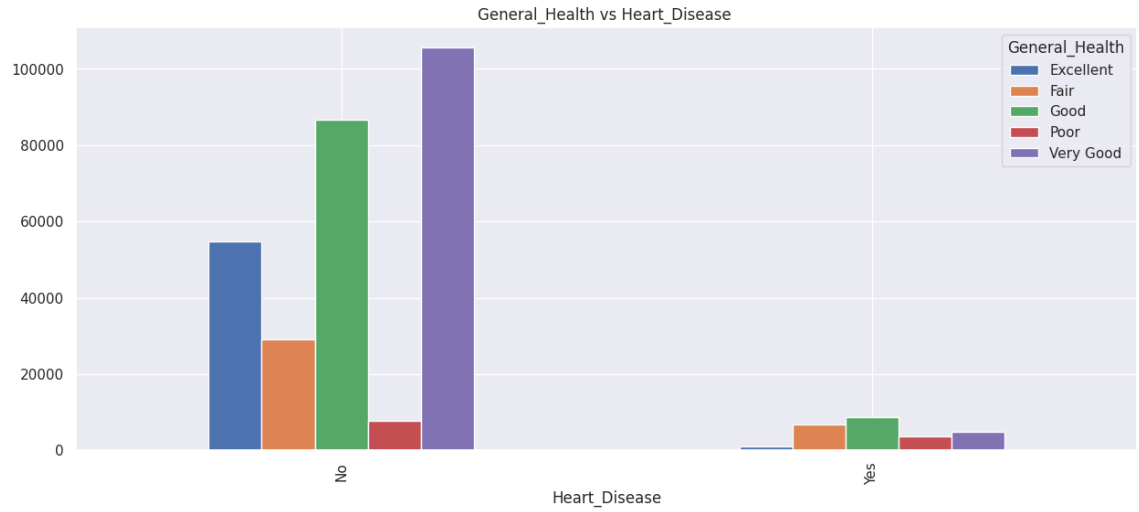


In [103]:

```
Heart_Disease = pd.crosstab(data["Heart_Disease"],data["General_Health"])
Heart_Disease.plot(kind="bar")
plt.title("General_Health vs Heart_Disease ")
```

Out[103]:

```
Text(0.5, 1.0, 'General_Health vs Heart_Disease ')
```

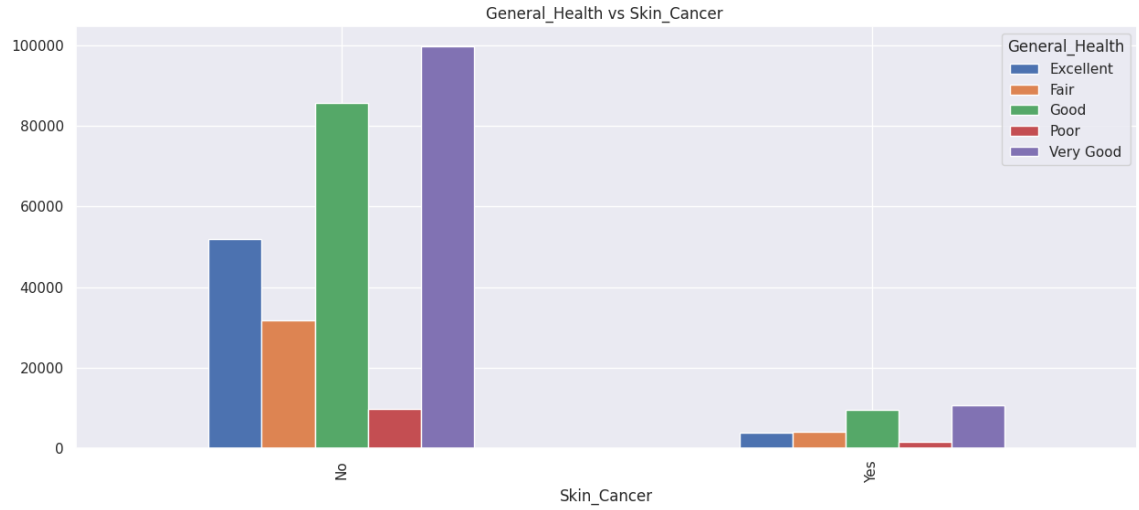


In [104]:

```
Skin_Cancer = pd.crosstab(data["Skin_Cancer"],data["General_Health"])
Skin_Cancer.plot(kind="bar")
plt.title("General_Health vs Skin_Cancer ")
```

Out[104]:

```
Text(0.5, 1.0, 'General_Health vs Skin_Cancer ')
```

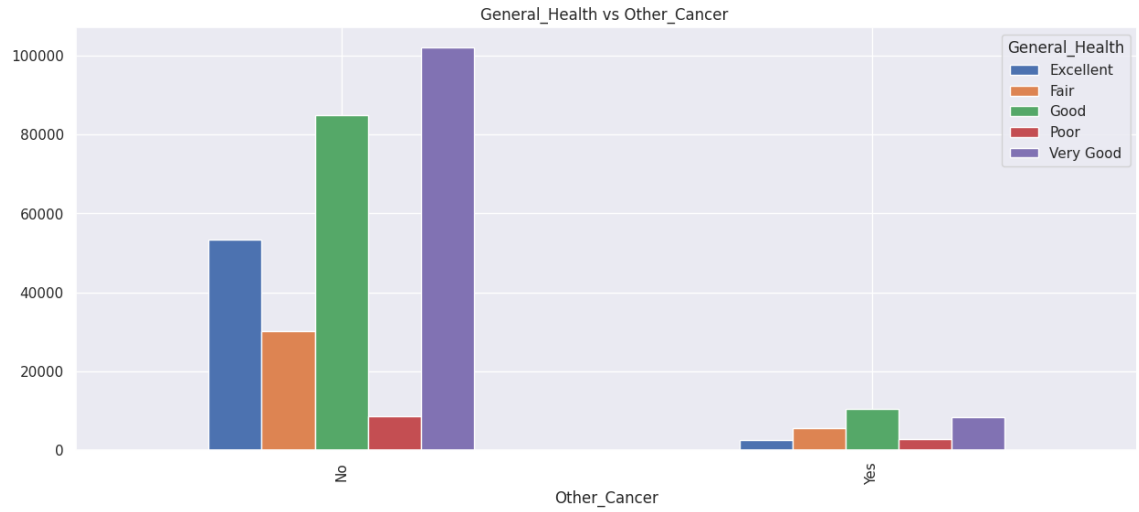


In [105]:

```
Other_Cancer = pd.crosstab(data["Other_Cancer"],data["General_Health"])
Other_Cancer.plot(kind="bar")
plt.title("General_Health vs Other_Cancer ")
```

Out[105]:

```
Text(0.5, 1.0, 'General_Health vs Other_Cancer ')
```

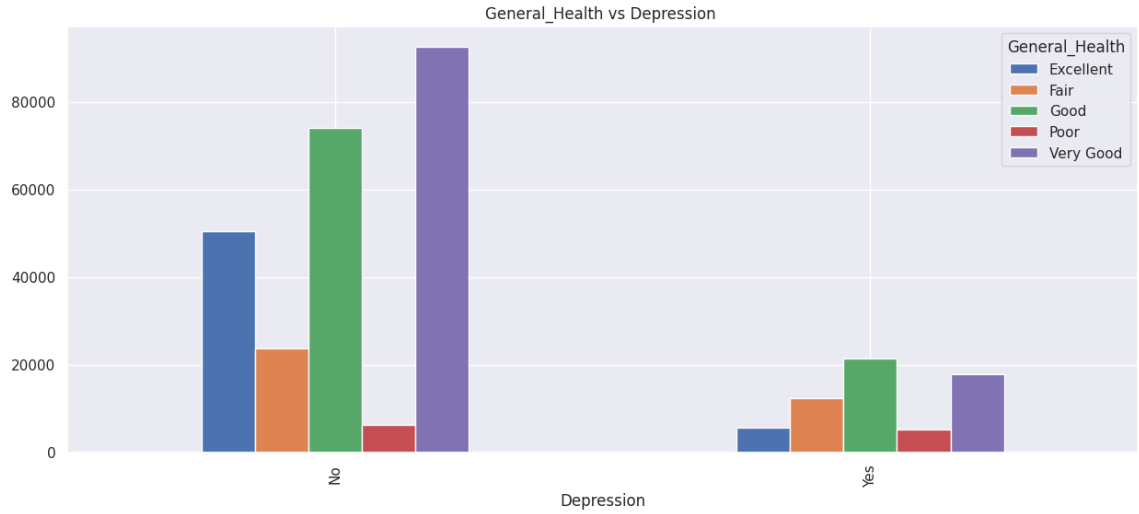


In [106]:

```
Depression = pd.crosstab(data["Depression"],data["General_Health"])
Depression.plot(kind="bar")
plt.title("General_Health vs Depression ")
```

Out[106]:

```
Text(0.5, 1.0, 'General_Health vs Depression ')
```

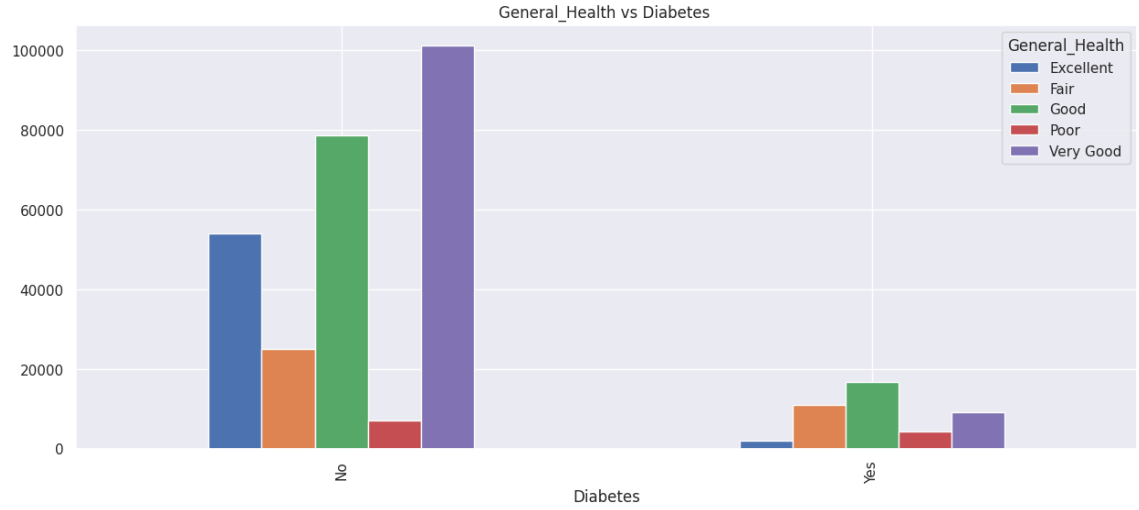


In [107]:

```
Diabetes = pd.crosstab(data["Diabetes"],data["General_Health"])
Diabetes.plot(kind="bar")
plt.title("General_Health vs Diabetes ")
```

Out[107]:

```
Text(0.5, 1.0, 'General_Health vs Diabetes ')
```

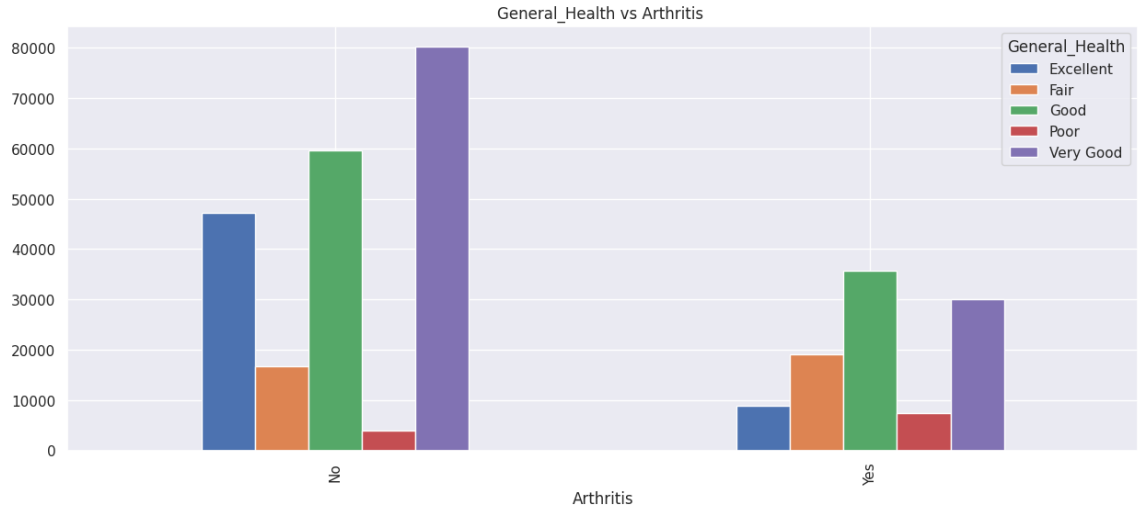


In [108]:

```
Arthritis = pd.crosstab(data["Arthritis"],data["General_Health"])
Arthritis.plot(kind="bar")
plt.title("General_Health vs Arthritis ")
```

Out[108]:

```
Text(0.5, 1.0, 'General_Health vs Arthritis ')
```

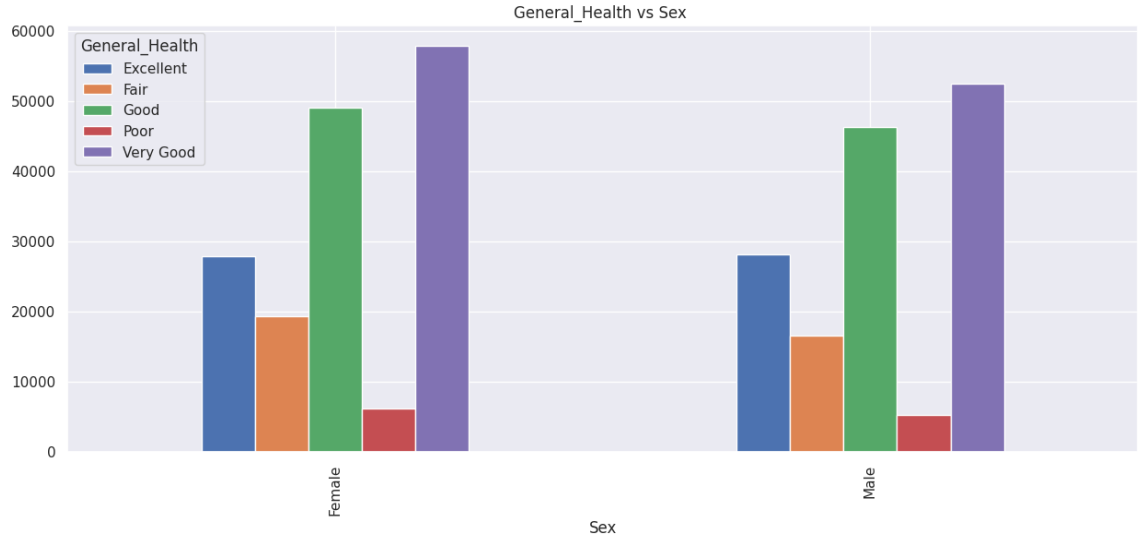


In [109]:

```
Sex = pd.crosstab(data["Sex"],data["General_Health"])
Sex.plot(kind="bar")
plt.title("General_Health vs Sex ")
```

Out[109]:

```
Text(0.5, 1.0, 'General_Health vs Sex ')
```

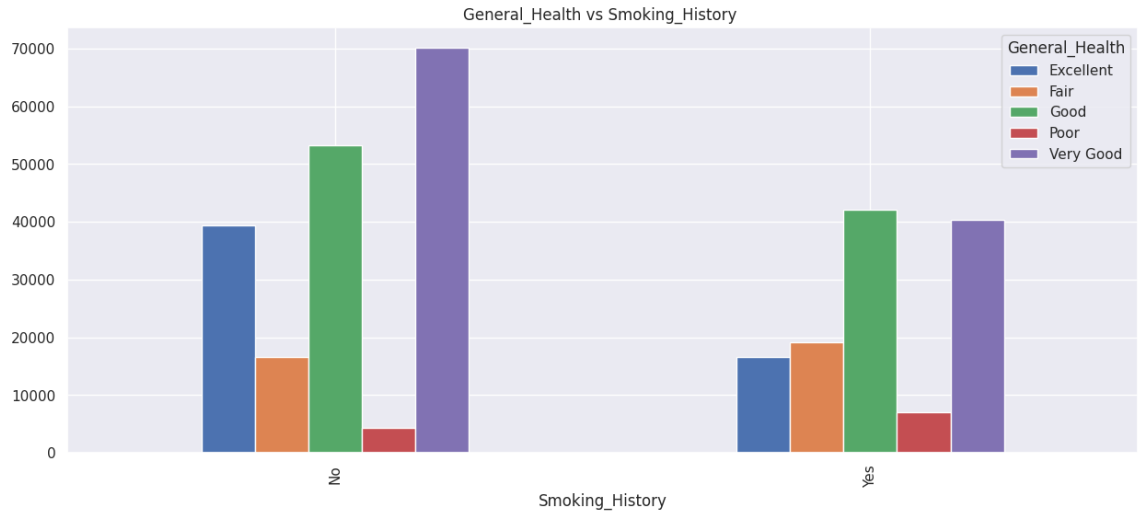


In [114]:

```
Smoking_History = pd.crosstab(data["Smoking_History"],data["General_Health"])
Smoking_History.plot(kind="bar")
plt.title("General_Health vs Smoking_History ")
```

Out[114]:

```
Text(0.5, 1.0, 'General_Health vs Smoking_History ')
```



In [116]:

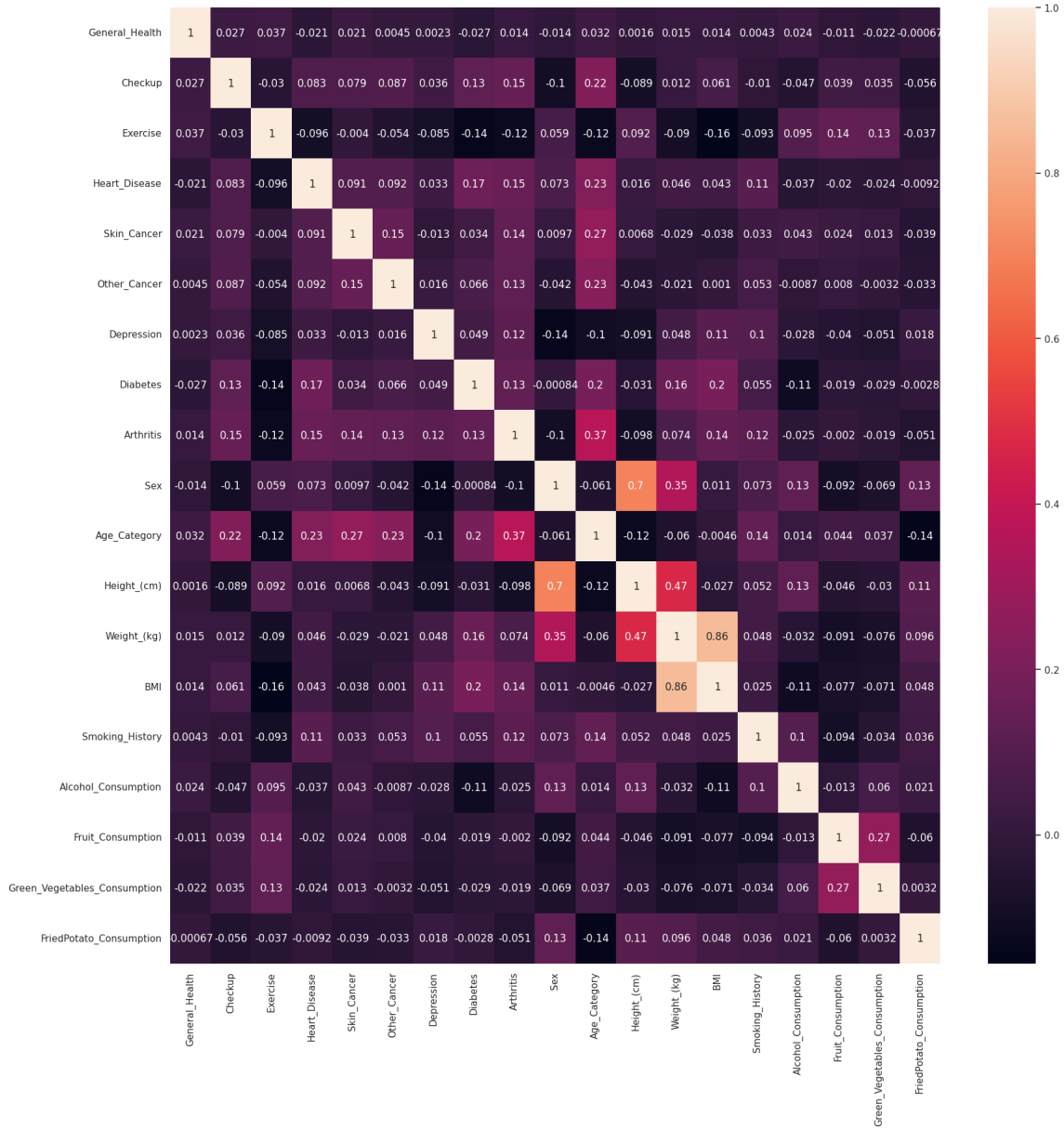
```
from sklearn.preprocessing import LabelEncoder ## using lable encoding on catigorical da
lable = LabelEncoder()

for i in catigorical_features:
  lable.fit(data[i])
  data[i] = lable.fit_transform(data[i])
```

In [122]:

```
plt.figure(figsize=(20,20))
sns.heatmap(data.corr(),annot=True)
```

Out[122]:

<Axes: >



In [123]:

```
data
```

Out[123]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Dep
0	3	2	0	0	0	0	
1	4	4	0	1	0	0	
2	4	4	1	0	0	0	
3	3	4	1	1	0	0	
4	2	4	0	0	0	0	
...	...	...	...	...	...	...	
308849	4	4	1	0	0	0	
308850	1	3	1	0	0	0	
308851	4	0	1	0	0	0	
308852	4	4	1	0	0	0	
308853	0	4	1	0	0	0	

308854 rows × 19 columns

In [143]:

```
# saprate dependent and indipendent features
x = data.drop("General_Health",axis=1)
y = data["Checkup"]
```

In [144]:

```
# now we have to split numerical and catigorical columns
catigorical_features = x.select_dtypes(include="object").columns
numerical_features = x.select_dtypes(exclude="object").columns
print(catigorical_features)
print("="*100)
print(numerical_features)
```

```
Index([], dtype='object')
====================================================================================
==========================
Index(['Checkup', 'Exercise', 'Heart_Disease', 'Skin_Cancer', 'Other_Cance
r',
       'Depression', 'Diabetes', 'Arthritis', 'Sex', 'Age_Category',
       'Height_(cm)', 'Weight_(kg)', 'BMI', 'Smoking_History',
       'Alcohol_Consumption', 'Fruit_Consumption',
       'Green_Vegetables_Consumption', 'FriedPotato_Consumption'],
      dtype='object')
```

In [145]:

```
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
```

In [146]:

```
# create numeric pipline
numpipline = Pipeline(
    steps=[
        ("imputer",SimpleImputer(strategy="median")),
        ("scaler",MinMaxScaler())
    ]
)

# Creating preprocessing object
preprocessor = ColumnTransformer([
    ("numpipline",numpipline,numerical_features)

])
```

In [147]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=42)
```

In [148]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(247083, 18)
(61771, 18)
(247083,)
(61771,)
```

In [152]:

```
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)
```

In [154]:

```

```

In [155]:

```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense ## Help To create Hidden Layers
```

In [158]:

```
model = Sequential()
model.add(Dense(18,activation="relu",input_dim=18))
model.add(Dense(300,activation="relu"))
model.add(Dense(250,activation="relu"))
model.add(Dense(150,activation="relu"))
model.add(Dense(5,activation="softmax"))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 18)                342

 dense_6 (Dense)             (None, 300)               5700

 dense_7 (Dense)             (None, 250)               75250

 dense_8 (Dense)             (None, 150)               37650

 dense_9 (Dense)             (None, 5)                 755

=================================================================
Total params: 119,697
Trainable params: 119,697
Non-trainable params: 0
_________________________________________________________________
```

In [159]:

```
model.layers
```

Out[159]:

```
[<keras.layers.core.dense.Dense at 0x7c8fa51fc3d0>,
 <keras.layers.core.dense.Dense at 0x7c8fa43902e0>,
 <keras.layers.core.dense.Dense at 0x7c8fa43915a0>,
 <keras.layers.core.dense.Dense at 0x7c8fa4391810>,
 <keras.layers.core.dense.Dense at 0x7c8fa4392ec0>]
```

In [160]:

```
LOSS_FUNCTION = "sparse_categorical_crossentropy"
OPTIMIZER = "Adam"
METRICES = ["accuracy"]
```

In [161]:

```
import time

def get_log_path(log_dir="logs/fit"):
  fileName = time.strftime("log_%Y_%m_%d_%H_%M_%S")
  logs_path = os.path.join(log_dir, fileName)
  print(f"Saving logs at {logs_path}")
  return logs_path

log_dir = get_log_path()
tb_cb = tensorflow.keras.callbacks.TensorBoard(log_dir=log_dir)
```

```
Saving logs at logs/fit/log_2023_08_02_13_19_18
```

In [162]:

```
## Early stoping
early_stopping = tensorflow.keras.callbacks.EarlyStopping(patience=5,restore_best_weight
```

In [163]:

```
# check point callback
CKPT_path = "Model_ckpt.h5"
checkpointing_cb = tensorflow.keras.callbacks.ModelCheckpoint(CKPT_path, save_best_only=
```

In [165]:

```
model.compile(optimizer=OPTIMIZER,loss=LOSS_FUNCTION,metrics=METRICES)
```

In [167]:

```
history = model.fit(X_train,y_train,epochs=50,validation_split=0.2,callbacks=[tb_cb,earl
```

```
Epoch 1/50
6178/6178 [==============================] - 27s 4ms/step - loss: 0.0014 -
accuracy: 0.9997 - val_loss: 2.3031e-05 - val_accuracy: 1.0000
Epoch 2/50
6178/6178 [==============================] - 30s 5ms/step - loss: 5.9695e-
04 - accuracy: 0.9999 - val_loss: 4.1927e-05 - val_accuracy: 1.0000
Epoch 3/50
6178/6178 [==============================] - 29s 5ms/step - loss: 1.6923e-
06 - accuracy: 1.0000 - val_loss: 1.3918e-06 - val_accuracy: 1.0000
Epoch 4/50
6178/6178 [==============================] - 30s 5ms/step - loss: 7.9161e-
04 - accuracy: 0.9999 - val_loss: 0.0019 - val_accuracy: 0.9995
Epoch 5/50
6178/6178 [==============================] - 31s 5ms/step - loss: 2.9649e-
04 - accuracy: 0.9999 - val_loss: 2.8628e-07 - val_accuracy: 1.0000
Epoch 6/50
6178/6178 [==============================] - 28s 5ms/step - loss: 3.1519e-
08 - accuracy: 1.0000 - val_loss: 2.8178e-08 - val_accuracy: 1.0000
Epoch 7/50
6178/6178 [==============================] - 30s 5ms/step - loss: 2.3110e-
09 - accuracy: 1.0000 - val_loss: 8.5795e-09 - val_accuracy: 1.0000
Epoch 8/50
6178/6178 [==============================] - 29s 5ms/step - loss: 1.4655e-
10 - accuracy: 1.0000 - val_loss: 4.0981e-09 - val_accuracy: 1.0000
Epoch 9/50
6178/6178 [==============================] - 29s 5ms/step - loss: 1.9902e-
11 - accuracy: 1.0000 - val_loss: 8.3706e-10 - val_accuracy: 1.0000
Epoch 10/50
6178/6178 [==============================] - 31s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 3.1360e-11 - val_accuracy: 1.0000
Epoch 11/50
6178/6178 [==============================] - 31s 5ms/step - loss: 6.0308e-
13 - accuracy: 1.0000 - val_loss: 1.2062e-11 - val_accuracy: 1.0000
Epoch 12/50
6178/6178 [==============================] - 30s 5ms/step - loss: 1.2062e-
12 - accuracy: 1.0000 - val_loss: 1.4474e-11 - val_accuracy: 1.0000
Epoch 13/50
6178/6178 [==============================] - 30s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 9.6493e-12 - val_accuracy: 1.0000
Epoch 14/50
6178/6178 [==============================] - 28s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 4.8246e-12 - val_accuracy: 1.0000
Epoch 15/50
6178/6178 [==============================] - 28s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 2.4123e-12 - val_accuracy: 1.0000
Epoch 16/50
6178/6178 [==============================] - 29s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 2.4123e-12 - val_accuracy: 1.0000
Epoch 17/50
6178/6178 [==============================] - 27s 4ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 2.4123e-12 - val_accuracy: 1.0000
Epoch 18/50
6178/6178 [==============================] - 27s 4ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 2.4123e-12 - val_accuracy: 1.0000
Epoch 19/50
6178/6178 [==============================] - 28s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 20/50
6178/6178 [==============================] - 30s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 21/50
```

```
6178/6178 [==============================] - 29s 5ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 22/50
6178/6178 [==============================] - 27s 4ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 23/50
6178/6178 [==============================] - 27s 4ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 24/50
6178/6178 [==============================] - 27s 4ms/step - loss: 0.0000e+
00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

In [171]:

```
from sklearn.metrics import accuracy_score,classification_report,roc_auc_score
```

In [173]:

```
ypredict = model.predict(X_test)
```

```
1931/1931 [==============================] - 5s 2ms/step
```

In [175]:

```
ypredict = ypredict.argmax(axis=-1)
```

In [184]:

```
print(accuracy_score(ypredict,y_test))
print(classification_report(ypredict,y_test))
```

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2634
           1       1.00      1.00      1.00       287
           2       1.00      1.00      1.00      7433
           3       1.00      1.00      1.00      3500
           4       1.00      1.00      1.00     47917

    accuracy                           1.00     61771
   macro avg       1.00      1.00      1.00     61771
weighted avg       1.00      1.00      1.00     61771
```

In [187]:

```
pd.DataFrame(history.history)
```

Out[187]:

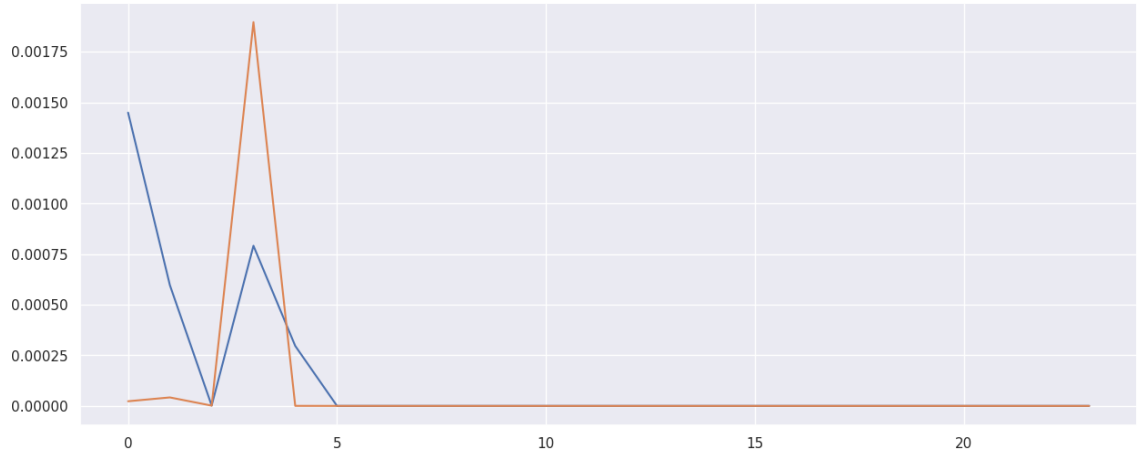
	loss	accuracy	val_loss	val_accuracy
0	1.449485e-03	0.999707	2.303125e-05	1.000000
1	5.969467e-04	0.999853	4.192749e-05	0.999980
2	1.692252e-06	1.000000	1.391809e-06	1.000000
3	7.916085e-04	0.999894	1.896595e-03	0.999494
4	2.964933e-04	0.999939	2.862804e-07	1.000000
5	3.151875e-08	1.000000	2.817765e-08	1.000000
6	2.311011e-09	1.000000	8.579503e-09	1.000000
7	1.465495e-10	1.000000	4.098113e-09	1.000000
8	1.990179e-11	1.000000	8.370558e-10	1.000000
9	0.000000e+00	1.000000	3.136005e-11	1.000000
10	6.030844e-13	1.000000	1.206156e-11	1.000000
11	1.206169e-12	1.000000	1.447388e-11	1.000000
12	0.000000e+00	1.000000	9.649251e-12	1.000000
13	0.000000e+00	1.000000	4.824626e-12	1.000000
14	0.000000e+00	1.000000	2.412313e-12	1.000000
15	0.000000e+00	1.000000	2.412313e-12	1.000000
16	0.000000e+00	1.000000	2.412313e-12	1.000000
17	0.000000e+00	1.000000	2.412313e-12	1.000000
18	0.000000e+00	1.000000	0.000000e+00	1.000000
19	0.000000e+00	1.000000	0.000000e+00	1.000000
20	0.000000e+00	1.000000	0.000000e+00	1.000000
21	0.000000e+00	1.000000	0.000000e+00	1.000000
22	0.000000e+00	1.000000	0.000000e+00	1.000000
23	0.000000e+00	1.000000	0.000000e+00	1.000000

In [188]:

```
# plot and see the accuracy and loss
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
```

Out[188]:

```
[<matplotlib.lines.Line2D at 0x7c8f0d7a0100>]
```

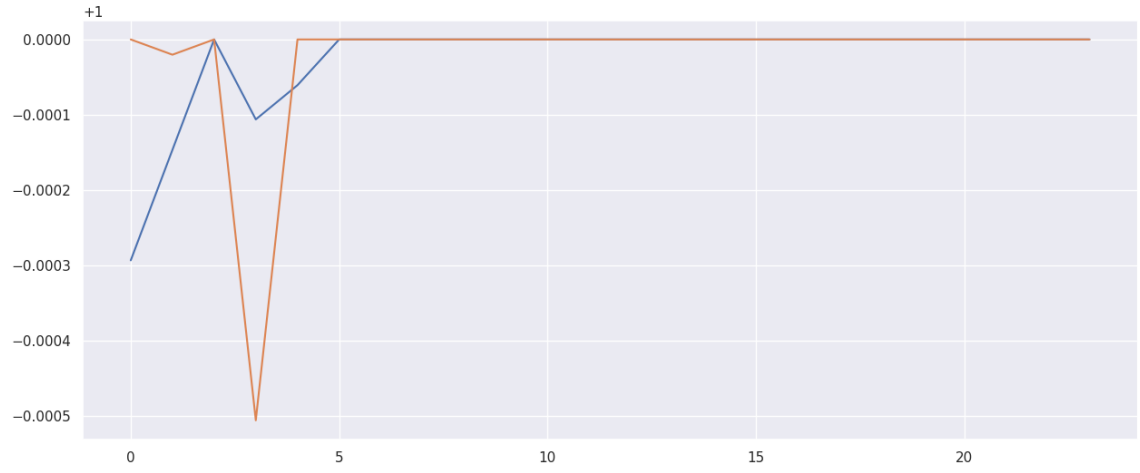


In [190]:

```
plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
```

Out[190]:

```
[<matplotlib.lines.Line2D at 0x7c8ef8e2ab90>]
```

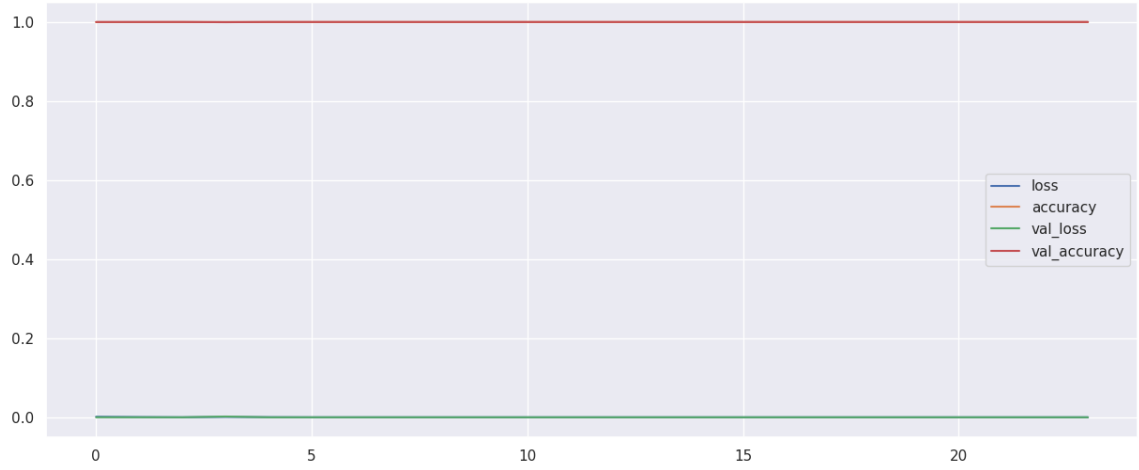


In [191]:

```
pd.DataFrame(history.history).plot()
```

Out[191]:

```
<Axes: >
```



In [192]:

```
y_test[:20]
```

Out[192]:

```
302051    4
59950     4
203639    4
78768     4
216156    4
84107     4
284341    4
227575    4
19010     2
123471    3
153765    4
197515    4
64189     0
152430    0
11518     2
252630    4
218232    4
231288    4
16249     1
244775    4
Name: Checkup, dtype: int64
```

In [193]:

```
ypredict[:20]
```

Out[193]:

```
array([4, 4, 4, 4, 4, 4, 4, 4, 2, 3, 4, 4, 0, 0, 2, 4, 4, 4, 1, 4])
```

In [ ]:

```
```