0.) Import and Clean data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
from sklearn.preprocessing import StandardScaler

import seaborn as sns
from sklearn.decomposition import PCA
```

In [3]:
```
df = pd.read_csv("Country-data.csv", sep = ",")
```

In [4]:
```
df.head()
```

Out[4]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13

In [5]:
```
df.columns
```

Out[5]:
```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

In [6]:
```
names = df[["country"]]
X = df.drop(["country"], axis = 1)


```

In [7]:
```
scaler = StandardScaler().fit(X)
X_scaled = scaler.transform(X)
```

In [8]:
```

```

In [8]:
```
1
```

1.) Run a PCA Algorithm to get 2 Principle Components for the 9 X features

In [12]:

```
pca = PCA(n_components = 2)
X_pca = pca.fit_transform(X_scaled)
X_pca
```

Out[12]:
```
array([[-2.91302459e+00,  9.56205755e-02],
       [ 4.29911330e-01, -5.88155666e-01],
       [-2.85225077e-01, -4.55174413e-01],
       [-2.93242265e+00,  1.69555507e+00],
       [ 1.03357587e+00,  1.36658709e-01],
       [ 2.24072616e-02, -1.77918658e+00],
       [-1.01583737e-01, -5.68251724e-01],
       [ 2.34216461e+00, -1.98845915e+00],
       [ 2.97376366e+00, -7.34688659e-01],
       [-1.81486997e-01, -4.02865873e-01],
       [ 1.26874386e+00, -6.56588363e-01],
       [ 1.67099640e+00,  5.61162493e-01],
       [-1.12385093e+00, -9.61397405e-01],
       [ 1.08137420e+00, -4.81969530e-01],
       [ 5.80025152e-01,  5.35326834e-01],
       [ 3.14378596e+00,  6.63547921e-01],
       [ 2.11255447e-01,  6.99242662e-01],
       [-2.67231388e+00,  4.18172125e-01],
       [-1.56570962e-01,  7.77395617e-01],
       [-7.93851561e-01, -1.20261085e-01],
       [ 9.95867143e-01, -9.71888439e-01],
       [-8.82087639e-01,  4.57368180e-01],
       [ 1.40781361e-01, -2.15107731e+00],
       [ 2.46008609e+00,  1.64540436e-02],
       [ 9.06594515e-01,  3.02776054e-02],
       [-3.12205344e+00,  3.87749688e-02],
       [-2.89897068e+00, -4.22663328e-01],
       [-5.82411867e-01,  8.94820332e-01],
       [-2.80790857e+00,  7.86488969e-02],
       [ 2.54363055e+00, -1.72709470e+00],
       [-1.55801452e-01,  3.51235458e-01],
       [-3.96496402e+00,  3.86619319e-01],
       [-3.55755520e+00,  1.28912809e+00],
       [ 9.51656055e-01, -1.07642827e+00],
       [ 5.74819803e-02, -1.18999652e+00],
       [ 1.21146120e-01, -1.76890914e+00],
       [-2.09355643e+00,  3.43600988e-01],
       [-3.17337012e+00,  1.05038163e+00],
       [-1.72567641e+00,  2.17634895e+00],
       [ 9.37826615e-01, -1.35047238e+00],
       [-2.58170623e+00,  1.20787342e+00],
       [ 1.14886344e+00, -8.44812046e-01],
       [ 2.17445492e+00, -4.51044737e-03],
       [ 2.05326329e+00,  4.23198280e-01],
       [ 3.01049182e+00, -8.65548729e-01],
       [-2.31102923e-01, -8.80641302e-01],
       [ 9.61833240e-03, -1.04522097e+00],
       [-8.48186699e-01, -8.19818902e-01],
       [ 8.18678445e-02, -5.67803943e-01],
       [-1.29342284e+00,  2.36369455e+00],
       [-2.47469590e+00, -6.18025236e-01],
       [ 1.65908340e+00,  1.02156447e+00],
       [-1.88828409e-01,  1.07176458e+00],
       [ 2.45896019e+00, -1.07614294e+00],
       [ 2.25427080e+00, -1.86663813e+00],
       [-1.42171455e+00,  3.19723358e-01],
       [-2.21366958e+00,  2.23495896e-01],
```

```
[ 3.21942207e-01, -5.18255225e-01],
[ 2.67142195e+00, -1.27360990e+00],
[-2.05416693e+00,  3.80034393e-01],
[ 1.77949294e+00, -1.76539693e+00],
[ 1.45504799e-01, -4.31336366e-01],
[-6.63503125e-01, -6.13910837e-01],
[-2.96952947e+00,  7.28533786e-01],
[-2.83361647e+00, -9.11281950e-02],
[-3.22781465e-01,  1.36134136e+00],
[-4.40971727e+00,  1.74223049e+00],
[ 1.83916013e+00,  1.27296493e+00],
[ 2.48092396e+00, -6.34701926e-01],
[-1.34282579e+00, -5.35138946e-01],
[-9.54750124e-01, -7.32361786e-01],
[-1.06461193e-03, -1.33434959e+00],
[-1.02922816e+00, -2.83269323e-01],
[ 3.66862804e+00,  1.72949317e+00],
[ 1.48531666e+00, -1.04922436e+00],
[ 2.16580995e+00, -1.77248548e+00],
[ 1.86093002e-02, -2.38961304e-01],
[ 2.26588199e+00, -2.43559383e+00],
[ 1.60142643e-01,  5.41065172e-01],
[-2.93346500e-01, -2.37525434e-01],
[-1.87470247e+00, -1.71029967e-01],
[-1.23921686e+00,  3.69138411e-01],
[ 2.46565870e+00,  8.80497785e-02],
[-3.39969880e-01,  1.29819641e+00],
[-1.52776995e+00,  5.45786891e-01],
[ 1.18883984e+00,  1.62040035e-01],
[ 1.17199076e+00, -2.56295112e-01],
[-1.80315140e+00,  2.03785098e+00],
[-1.77358023e+00,  1.05339867e+00],
[ 8.18943051e-01,  3.89841660e-01],
[ 1.40978812e+00,  7.29833198e-01],
[ 6.91775496e+00,  4.84984369e+00],
[ 7.33210319e-01, -9.48674314e-02],
[-2.13600867e+00,  3.42733042e-01],
[-2.97988525e+00,  2.16622419e-01],
[ 1.23082842e+00,  1.60174864e+00],
[ 1.10860101e+00,  1.00931426e+00],
[-3.41225513e+00,  5.61468514e-01],
[ 3.67954260e+00,  4.76548605e+00],
[-1.95392747e+00,  1.38338452e+00],
[ 8.99775055e-01,  4.16479781e-01],
[-3.80928795e-01,  1.01773629e-01],
[ 5.09539453e-01,  1.61658340e-01],
[-9.44975538e-01,  5.29799562e-01],
[ 1.02668389e+00, -2.57641566e-01],
[-2.32870156e-01, -2.81027769e-01],
[-2.92054051e+00,  8.93270294e-01],
[-1.83719774e+00, -1.61366899e+00],
[-1.04337471e+00,  1.00284112e+00],
[-1.30708985e+00, -7.89048631e-01],
[ 3.37915727e+00,  1.15702442e-01],
[ 1.81574666e+00, -1.58472369e+00],
[-3.45016774e+00,  9.69922452e-01],
[-4.91206615e+00, -9.44986846e-02],
```

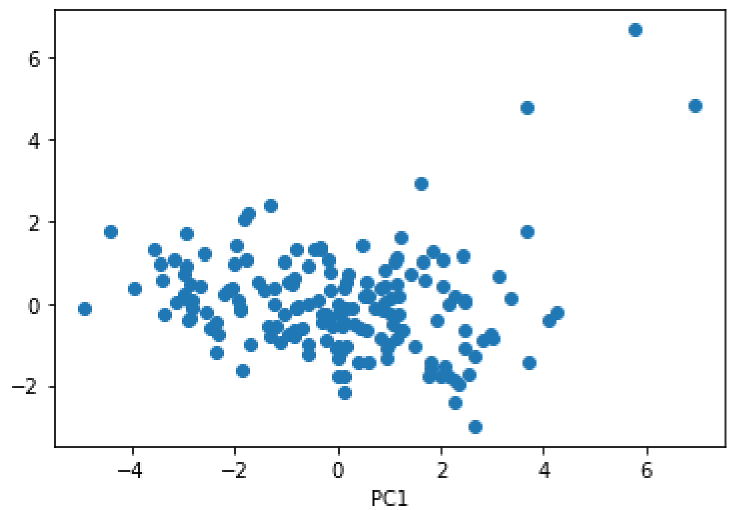
```
[ 3.72119513e+00, -1.44725498e+00],
[ 1.12738665e+00,  4.91611136e-01],
[-2.36034718e+00, -4.79399646e-01],
[ 1.16378429e+00,  1.11527620e+00],
[ 1.17846224e-01,  3.61031140e-01],
[-2.06354519e-02, -1.08661741e+00],
[-7.82745871e-01, -9.64980905e-02],
[ 1.21782754e+00, -6.59168961e-01],
[ 1.81406748e+00, -1.45088654e+00],
[ 4.24229634e+00, -1.95603674e-01],
[ 5.72792704e-01, -6.37384843e-01],
[ 1.63761544e-01, -1.06667848e+00],
[-1.67970356e+00, -1.00162862e+00],
[-5.62897632e-01, -2.21043960e-02],
[ 8.55935813e-01, -1.83440759e-01],
[-1.91217031e+00,  9.15599347e-02],
[ 8.32420187e-01, -8.69325996e-01],
[ 1.60259775e+00,  2.93912057e+00],
[-3.38162479e+00, -2.36301516e-01],
[ 5.78337630e+00,  6.68209028e+00],
[ 2.02972370e+00,  1.05040745e+00],
[ 2.27949171e+00,  1.95275226e-01],
[-8.06209136e-01,  1.30349059e+00],
[-1.19183736e+00, -5.56757164e-01],
[ 1.91806245e+00, -4.27468245e-01],
[ 2.01919721e+00, -1.78438246e+00],
[-5.75572155e-01, -9.97551478e-01],
[ 2.66234652e-02, -1.60640815e-02],
[-2.31942387e+00, -7.69407328e-01],
[ 1.71674731e-01, -9.48076409e-02],
[ 2.81832286e+00, -9.14480968e-01],
[ 4.08854413e+00, -4.29461909e-01],
[-1.24446436e+00, -2.89174316e-02],
[-2.55404919e+00, -2.15027956e-01],
[ 9.26092707e-01,  8.28230655e-01],
[-2.37197047e+00, -1.17751295e+00],
[-1.99764225e+00,  9.58361586e-01],
[-7.55008538e-01, -8.78938568e-02],
[ 6.02231612e-01,  1.73435708e-01],
[ 4.01437705e-01, -1.41198973e+00],
[-4.63936165e-01,  1.29187347e+00],
[-2.85483624e+00, -3.52082382e-01],
[ 3.02299800e-01, -9.75710669e-02],
[ 2.42714125e+00,  1.15181307e+00],
[ 2.06798993e+00, -1.53531349e+00],
[ 2.64120583e+00, -2.99736446e+00],
[ 6.17312598e-01, -1.43047723e+00],
[-8.53528944e-01, -6.54485112e-01],
[-8.20631131e-01,  6.39570072e-01],
[-5.51035564e-01, -1.23388618e+00],
[ 4.98524385e-01,  1.39074432e+00],
[-1.88745106e+00, -1.09453015e-01],
[-2.86406392e+00,  4.85997985e-01]])
```

In [35]:
```
```

In [35]:
```
```

2.) Plot a Scatter plot of the PCs on the axis

In [14]:
```
plt.scatter(x = X_pca[:,0], y = X_pca[:,1])
plt.xlabel("PC1")
plt.show()
```



In [36]:
```
```

3.) Rank the features in order of importance according to PCA

In [15]:
```
loadings = pca.components_
loadings
```

Out[15]:
```
array([[-0.41951945,  0.28389698,  0.15083782,  0.16148244,  0.39844111,
        -0.19317293,  0.42583938, -0.40372896,  0.39264482],
       [ 0.19288394,  0.61316349, -0.24308678,  0.67182064,  0.02253553,
        -0.00840447, -0.22270674,  0.15523311, -0.0460224 ]])
```

In [17]:
```
pd.DataFrame(np.sum(loadings**2, axis = 0))
```

Out[17]:

	0
0	0.213201
1	0.456567
2	0.081843
3	0.477420
4	0.159263
5	0.037386
6	0.230937
7	0.187094
8	0.156288

In [39]:
```
feature_importance = pd.DataFrame(np.sum(loadings**2, axis = 0))
```

In [40]:
```
feature_importance.index = feature_names
```

4.) Plot a heatmap of the feature importance (Fill in all parameters)

In [41]:
```
feature_names = df.columns[1:]
```

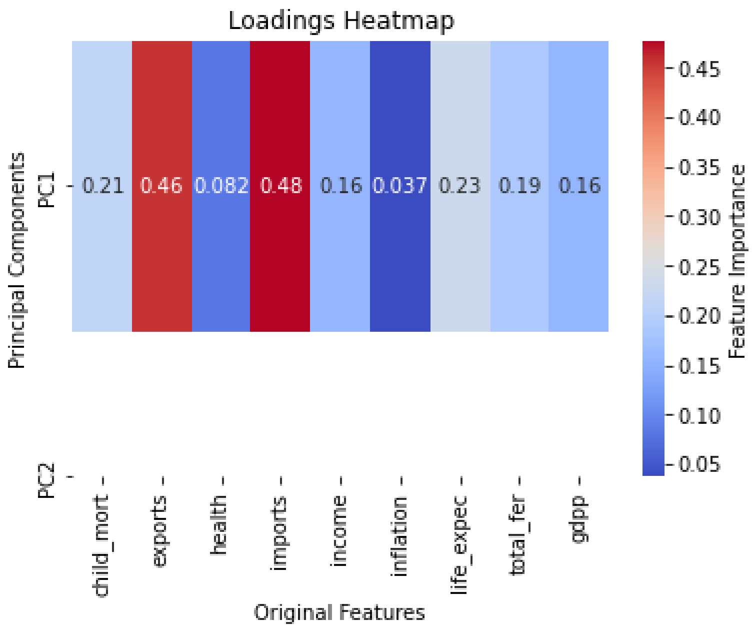
In [48]:

```

nnot= True, cmap='coolwarm', xticklabels= feature_names, yticklabels=["PC1'


)


```

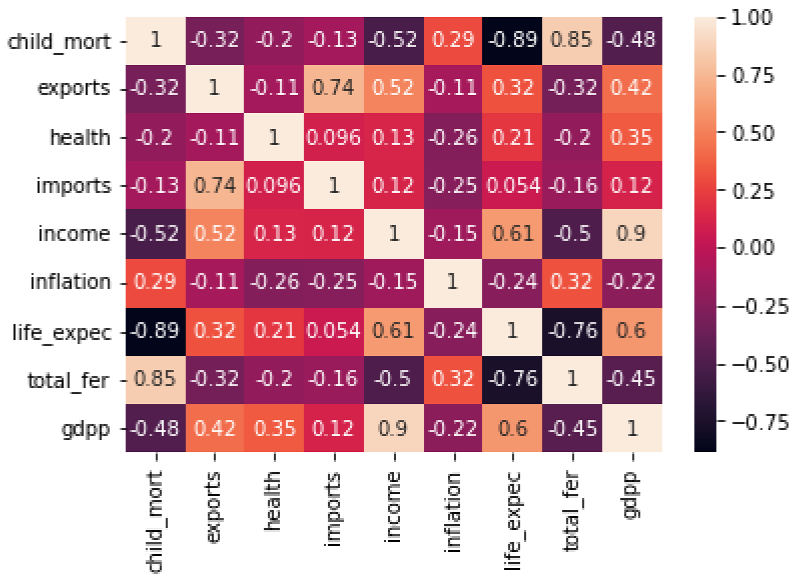


5.) Plot a correlation plot of the original features. What do you notice between the graphs of 4 & 5?

In [44]:

```
sns.heatmap(X.corr(), annot = True)
plt.plot()
```

Out[44]: []



In PC2, both imports and exports display high feature importance, indicating their significance in this component. Notably, these two variables exhibit a strong positive correlation of 0.74. As PCA was developed to remove correlation among variables, this suggests that the two highly correlated variables—imports and exports—share similar values within this particular component. The PCA process has successfully identified their underlying relationship and represented it in PC2.

On the other hand, PC1 highlights another interesting relationship: child mortality and total fertility. In this component, these two variables have large, negative values, yet they remain correlated. This demonstrates that even though their values are negative, there is still a meaningful association between the two. The PCA method effectively captures the highest correlations between features, and in this case, it has assigned similar principal component values to child mortality and total fertility in PC1.

In summary, the PCA analysis has successfully captured key relationships between variables in the dataset, condensing them into principal components that represent significant patterns. By analyzing these components, we can better understand the underlying structure of the data, which can help inform our decisions and insights.

6.) Run a PCA with 9 PCs. Plot a Cumulative Explained Variance Plot. How many PCs should we use if we want to retain 95% of the variance?

In [54]:

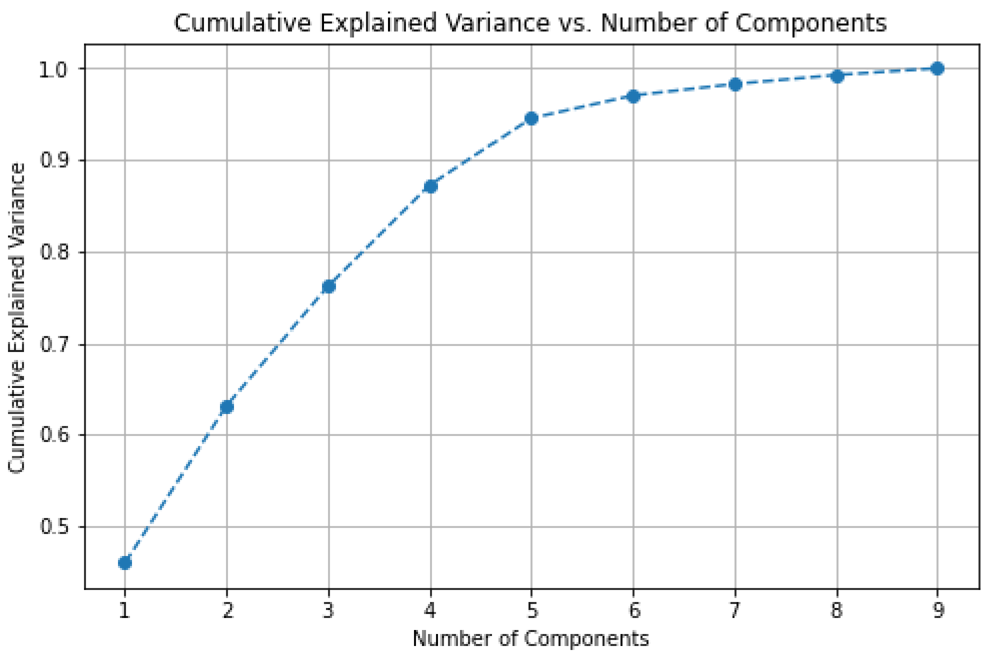
```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform PCA with 9 components
pca = PCA(n_components=9)
X_pca = pca.fit_transform(X_scaled)

# Calculate the cumulative explained variance
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratic

# Plot the cumulative explained variance
plt.figure(figsize=(8, 5))
plt.plot(range(1, 10), cumulative_explained_variance, marker='o', line
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance vs. Number of Components')
plt.grid()
plt.show()

# Determine the number of PCs needed to retain 95% of the variance
num_pcs = np.where(cumulative_explained_variance >= 0.95)[0][0] + 1
print(f"Number of PCs required to retain 95% of the variance: {num_pcs
```




```
Number of PCs required to retain 95% of the variance: 6
```

In [51]:

```
pca = PCA(n_components = 9)
X_pca = pca.fit_transform(X_scaled)
```

In [55]:

```
cumulative_explained_variance
```

Out[55]:

```
array([0.4595174 , 0.63133365, 0.76137624, 0.87190786, 0.94530998,
       0.97015232, 0.98275663, 0.99256944, 1.        ])
```

In [ ]:

```

```