In [10]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
```

## Data Preparation

In [16]:
```
crypto_df = pd.read_csv('crypto_data.csv')
crypto_df.head(3)
```

Out[16]:

| | Unnamed: 0 | CoinName | Algorithm | IsTrading | ProofType | TotalCoinsMined | TotalCoinSupply |
|---|---|---|---|---|---|---|---|
| 0 | 42 | 42 Coin | Scrypt | True | PoW/PoS | 4.199995e+01 | 42 |
| 1 | 365 | 365Coin | X11 | True | PoW/PoS | NaN | 2300000000 |
| 2 | 404 | 404Coin | Scrypt | True | PoW/PoS | 1.055185e+09 | 532000000 |

In [17]:
```
# Filter for Traded Crypto only and drop column
crypto_df = crypto_df[crypto_df['IsTrading'] == True]
crypto_df = crypto_df.drop(columns ='IsTrading')
```

In [18]:
```
# Remove rows with at least one null value
crypto_df = crypto_df.dropna()
# Filter for cryptocurrencies that have been mined
crypto_df = crypto_df[crypto_df["TotalCoinsMined"]> 0]
```

In [19]:
```
# Since the coin names do not contribute to the analysis of the data, delete the
crypto_df.drop(columns='CoinName', axis=1, inplace= True)
crypto_df.drop(columns='Unnamed: 0', axis=1, inplace= True)
crypto_df
```

Out[19]:

| | Algorithm | ProofType | TotalCoinsMined | TotalCoinSupply |
|---|---|---|---|---|
| 0 | Scrypt | PoW/PoS | 4.199995e+01 | 42 |
| 2 | Scrypt | PoW/PoS | 1.055185e+09 | 532000000 |
| 5 | X13 | PoW/PoS | 2.927942e+10 | 314159265359 |
| 7 | SHA-256 | PoW | 1.792718e+07 | 21000000 |
| 8 | Ethash | PoW | 1.076842e+08 | 0 |
| ... | ... | ... | ... | ... |
| 1238 | SHA-256 | DPoS | 2.000000e+09 | 2000000000 |
| 1242 | Scrypt | PoW/PoS | 1.493105e+07 | 250000000 |
| 1245 | CryptoNight | PoW | 9.802226e+08 | 1400222610 |

| | Algorithm | ProofType | TotalCoinsMined | TotalCoinSupply |
|---|---|---|---|---|
| **1246** | Equihash | PoW | 7.296538e+06 | 21000000 |
| **1247** | Scrypt | PoS | 1.283270e+05 | 1000000 |

532 rows × 4 columns

In [22]:
```
# Convert remaining features of test values to numerical data (algorithm and pro
X = pd.get_dummies(crypto_df, columns=['Algorithm', 'ProofType'])
X
```

Out[22]:

| | TotalCoinsMined | TotalCoinSupply | Algorithm_1GB AES Pattern Search | Algorithm_536 | Algorithm_Argon2d | Algo |
|---|---|---|---|---|---|---|
| **0** | 4.199995e+01 | 42 | 0 | 0 | 0 | |
| **2** | 1.055185e+09 | 532000000 | 0 | 0 | 0 | |
| **5** | 2.927942e+10 | 314159265359 | 0 | 0 | 0 | |
| **7** | 1.792718e+07 | 21000000 | 0 | 0 | 0 | |
| **8** | 1.076842e+08 | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | |
| **1238** | 2.000000e+09 | 2000000000 | 0 | 0 | 0 | |
| **1242** | 1.493105e+07 | 250000000 | 0 | 0 | 0 | |
| **1245** | 9.802226e+08 | 1400222610 | 0 | 0 | 0 | |
| **1246** | 7.296538e+06 | 21000000 | 0 | 0 | 0 | |
| **1247** | 1.283270e+05 | 1000000 | 0 | 0 | 0 | |

532 rows × 98 columns

In [23]:
```
# The dataframe has increased by 94 columns
# Standardize the dataset
scaler = StandardScaler()
crypto_scaled = scaler.fit_transform(X)
crypto_scaled
```

Out[23]:
```
array([[-0.11710817, -0.1528703 , -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [-0.09396955, -0.145009  , -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [ 0.52494561,  4.48942416, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       ...,
       [-0.09561336, -0.13217937, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [-0.11694817, -0.15255998, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [-0.11710536, -0.15285552, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ]])
```

# Dimensionality Reduction

## PCA

In [26]:
```
# preserve 90% of the explained variance in dimensionality reduction
pca = PCA(n_components=.90)
crypto_pca = pca.fit_transform(crypto_scaled)
df_crypto_pca = pd.DataFrame(data=crypto_pca)
df_crypto_pca.head()
# the number of features went from 98 to 74 principal components
```

Out[26]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | -0.335099 | 1.032189 | -0.590713 | 0.001397 | 8.903525e-15 | 3.715567e-12 | -1.716189e-14 | -0.007129 |
| **1** | -0.318434 | 1.032331 | -0.591126 | 0.001386 | 8.797456e-15 | 3.685202e-12 | -1.704862e-14 | -0.007739 |
| **2** | 2.305468 | 1.656383 | -0.683617 | 0.004731 | 1.292867e-14 | 1.259214e-11 | -2.400344e-14 | -0.054781 |
| **3** | -0.145184 | -1.320593 | 0.192813 | -0.001229 | -2.452485e-15 | -3.268574e-12 | 4.249357e-15 | -0.002071 |
| **4** | -0.151768 | -2.036192 | 0.396182 | -0.001705 | -1.119309e-14 | -4.534708e-12 | 1.170787e-14 | 0.027735 |

5 rows × 74 columns

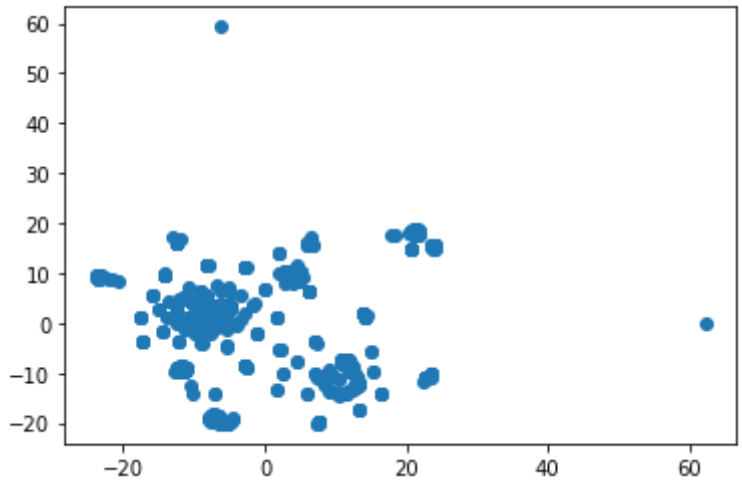
## t-SNE

In [27]:
```
tsne = TSNE(learning_rate=35)
```

In [29]:
```
tsne_features = tsne.fit_transform(df_crypto_pca)
tsne_features
```

Out[29]:
```
array([[ 10.864767 ,  -7.303208 ],
       [  9.183322 , -13.722738 ],
       [ 22.407055 , -11.618703 ],
       ...,
       [-23.372099 ,   9.55432  ],
       [-12.482249 ,  16.182966 ],
       [  3.4227908,   9.705258 ]], dtype=float32)
```

In [35]:
```
# TSNE reduces the data to have to principal components
# Create a scatterplot
x = tsne_features[:,0]
y = tsne_features[:,1]
plt.scatter(x, y)
plt.show()
# seem to be two general clusters between (-20,0) and (0, 20) but not very clear
```



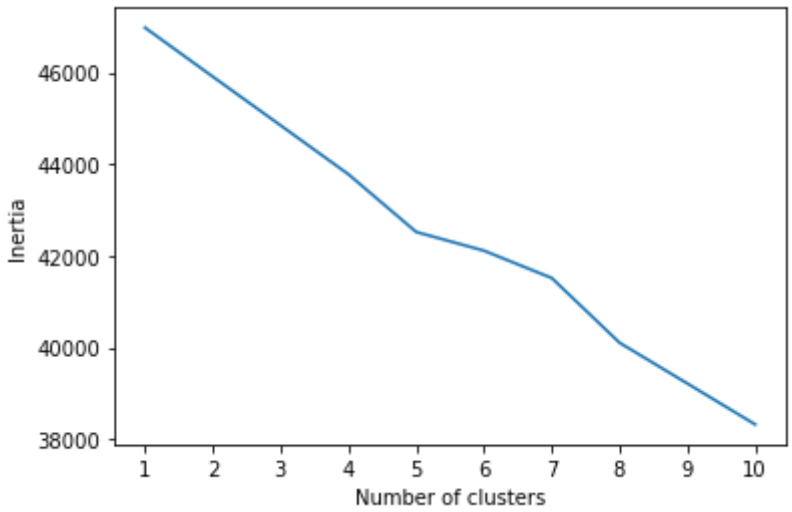

## Cluster Analysis with k-Means

In [38]:

```
# Finding the best value for k
inertia = []
k = list(range(1, 11))

# Calculate the inertia for the range of k values from 1-10
for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(df_crypto_pca)
    inertia.append(km.inertia_)

# Creating the Elbow Curve
elbow_data = {"k": k, "inertia": inertia}
df_elbow = pd.DataFrame(elbow_data)
plt.plot(df_elbow['k'], df_elbow['inertia'])
plt.xticks(range(1,11))
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
# no clear elbow
```




## Recommendation: Based on my findings, I cannot confidently state that the cryptocurrencies can be clustered together.

In [ ]: