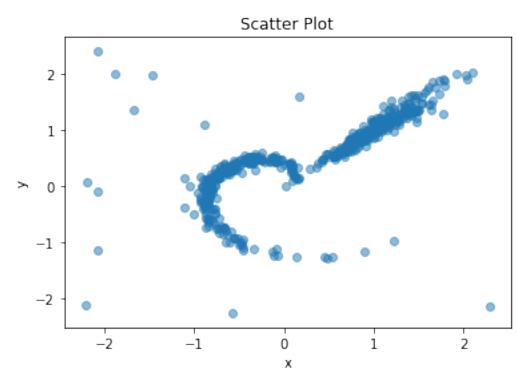
Step1: Read Data

In [3]:
```
import os, pandas as pd, numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
data = pd.read_csv(r'testData.csv', header = None)
data.columns = ['x', 'y']
data.describe()

plt.scatter(data['x'], data['y'], alpha=0.5)
plt.title('Scatter Plot')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Step 2: K-Means Clustering

In [5]:
```
# initiate
data['cluster'] = 0
center_1 = [1, 1]
center_2 = [0, 0]
```

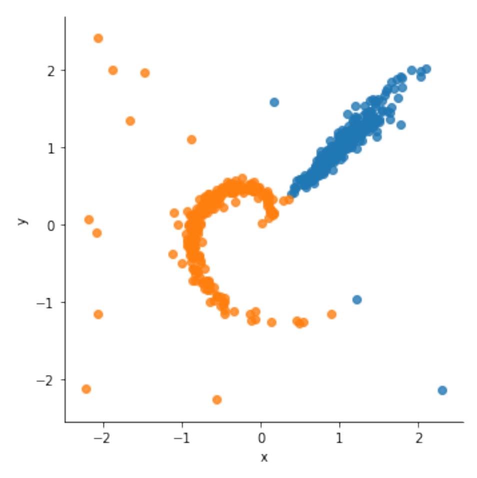
In [6]:
```
#define 2-step function
def assign(df):
    for index, p in df.iterrows():
        d1 = (p['x'] - center_1[0])**2 + (p['y'] - center_1[1])**2
        d2 = (p['x'] - center_2[0])**2 + (p['y'] - center_2[1])**2
        if d1 < d2:
            df.at[index,'cluster'] = 1
        else:
            df.at[index,'cluster'] = 2

def update(df, c_1, c_2):
    df_1 = df[df['cluster'] == 1]
    df_2 = df[df['cluster'] == 2]
    c_1 = [df_1['x'].mean(), df_1['y'].mean()]
    c_2 = [df_2['x'].mean(), df_2['y'].mean()]
    return df, c_1, c_2
```

In [7]:
```
#implement
move = 999
while move > 0:
    cluster_before = data['cluster'].copy()
    assign(data)
    data, center_1, center_2 = update(data, center_1, center_2)
    cluster_after = data['cluster'].copy()
    move = sum(abs(cluster_after - cluster_before))
```

In [8]:
```
#plot grouping
sns.lmplot(x = "x", y = "y", data = data, fit_reg = False, hue = 'cluster', legend = False)
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0xb452ba8>



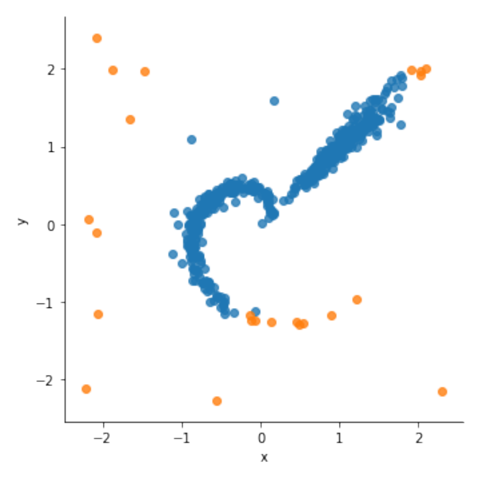
Step 3: Identify outliers

In [10]:
```
# Calculate Distance
data['d1'] = np.sqrt((data['x'] - center_1[0])**2 + (data['y'] - center_1[1])**2)
data['d2'] = np.sqrt((data['x'] - center_2[0])**2 + (data['y'] - center_2[1])**2)
data['dist'] = pd.DataFrame([data['d1'], data['d2']]).min()
```

In [11]:
```
# Calculate IQR
Q1 = np.percentile(data['dist'], 25)
Q3 = np.percentile(data['dist'], 75)
IQR = Q3 - Q1
low = Q1 - 1.5 * IQR
high = Q3 + 1.5 * IQR
```

In [12]:
```
data['outlier'] = 0
data.loc[(data['dist'] < low) | (data['dist'] > high), 'outlier'] = 1
sns.lmplot(x = "x", y = "y", data = data, fit_reg = False, hue = 'outlier', legend = False)
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x7a98b70>



Step 4: Group One

In [13]:
```
#%% Seperate groups
data_1 = data[data['cluster'] == 1].copy()
data_2 = data[data['cluster'] == 2].copy()
```

In [14]:
```
# Create X matrix and hat matrix
x = np.array([np.ones(len(data_1['x'])), data_1['x']])
x = x.T
x_inv = np.linalg.inv(np.dot(x.T, x))
hat_mat = np.dot(x, x_inv).dot(x.T)
hat_val = np.diagonal(hat_mat)
```

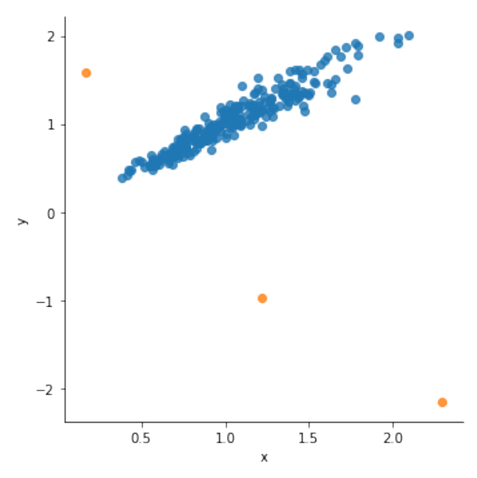
In [15]:
```
# Calculate residuals and MSE
y = np.array(data_1['y'])
y_hat = np.dot(hat_mat, y)
mse = sum((y_hat -y)**2)/(len(y)-2)
```

In [16]:
```
# Cook Distance
Cook_d = (y-y_hat)**2*hat_val/(1-hat_val)/(1-hat_val)/mse

cap = 4/(len(y)-2)
cap = 0.2
```

In [18]:
```
data_1['outlier'] = 0
data_1.loc[Cook_d > cap, 'outlier'] = 1
sns.lmplot(x = "x", y = "y", data = data_1, fit_reg = False, hue = 'outlier', legend = False)
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x7ac1c50>

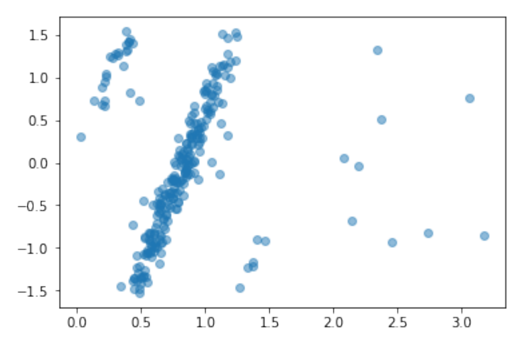


Step 5: Group Two

In [19]:
```
# Transformation
data_2['r'] = (data_2['x']**2 + data_2['y']**2)**0.5
data_2['theta'] = np.arctan(data_2['y']/data_2['x'])

plt.scatter(data_2['r'], data_2['theta'], alpha=0.5)
```

Out[19]: <matplotlib.collections.PathCollection at 0x7cab748>



In [20]:
```
# define functions for density based clustering
# iterate to expand the neighborhood
def DBSCAN(df, eps, MinPts):
    C = 0
    df_unvisit = df[df['visited'] == 0]
    while len(df_unvisit) > 0:
        for index, p in df_unvisit.iterrows():
            #print(index)
            if(df.loc[index, 'visited'] == 0):
                df.loc[index, 'visited'] = 1
                NeighborPts = regionQuery(df, df['r'], df['theta'], p['r'], p['theta'], eps)
                if len(NeighborPts) < MinPts:
                    df[index, 'cluster'] = 0
                else:
                    C += 1
                    new, NeighborPts = expandCluster(NeighborPts, NeighborPts, C, eps, MinPts, df)
                    while len(new) > 0:
                        df_iterate = df.loc[list(new)]
                        new, NeighborPts = expandCluster(df_iterate, NeighborPts, C, eps, MinPts, df)
                    i = NeighborPts[NeighborPts['cluster'] == C].index
                    df.loc[i, 'cluster'] = C
                    i = NeighborPts[NeighborPts['visited'] == 1].index
                    df.loc[i, 'visited'] = 1
        df_unvisit = df[df['visited'] == 0]
    return df
```

In [21]:
```
def expandCluster(p, NeighborPts, C, eps, MinPts, df):
    NeighborPts_old = NeighborPts
    for index, q in p.iterrows():
        NeighborPts.loc[index, 'visited'] = 1
        NeighborPts_add = regionQuery(df, df['r'], df['theta'], q['r'], q['theta'], eps)
        if len(NeighborPts_add) >= MinPts:
            NeighborPts = pd.concat([NeighborPts, NeighborPts_add])
        if q['cluster'] == 0:
            NeighborPts.loc[index, 'cluster'] = C
    NeighborPts.drop_duplicates(inplace=True)
    new = set(NeighborPts.index) - set(NeighborPts_old.index)
    return new, NeighborPts
```

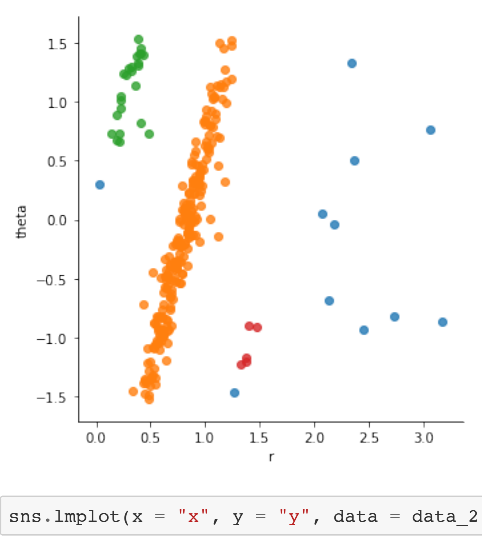
In [22]:
```
def regionQuery(df, df1, df2, p1, p2, eps):
    dist = np.sqrt((df1 - p1)**2 + (df2 - p2)**2)
    region = df[dist <= eps]
    return region
```

In [23]:
```
# initiate
data_2['cluster'] = 0
data_2['visited'] = 0
```

In [25]:
```
# Run DBSCAN
eps = 0.3
MinPts = 5
data_2 = DBSCAN(data_2, eps, MinPts)
```

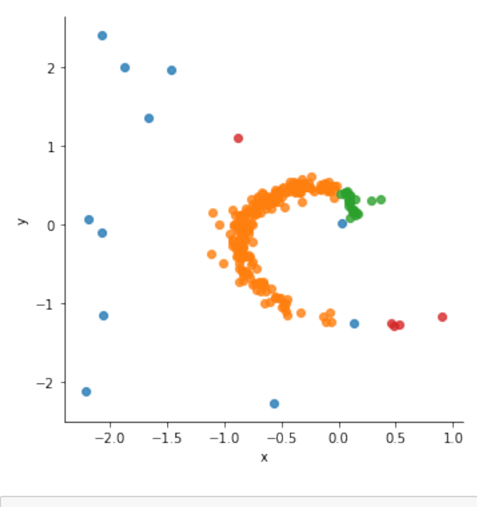
In [26]:
```
sns.lmplot(x = "r", y = "theta", data = data_2, fit_reg = False, hue = 'cluster', legend = False)
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x7ccfa20>



In [27]:
```
sns.lmplot(x = "x", y = "y", data = data_2, fit_reg = False, hue = 'cluster', legend = False)
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x7d4aa90>



In [28]:
```
data_2['outlier'] = 0
data_2.loc[data_2['cluster'] == 0,'outlier'] = 1
```

Step 6: Combine groups and output

In [29]:
```
data_1 = data_1[['x', 'y', 'outlier']]
data_2 = data_2[['x', 'y', 'outlier']]
data = pd.concat([data_1, data_2])

sns.lmplot(x = "x", y = "y", data = data, fit_reg = False, hue = 'outlier', legend = False)

data.to_csv('Quiz_out.csv')
```