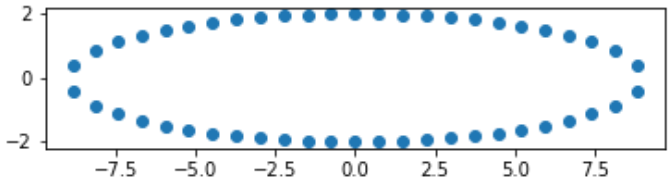
In [2]:

```
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import numpy as np
from sklearn.linear_model import SGDRegressor
import numpy as np
import scipy as sp
import scipy.optimize

def angles_in_ellipse(num,a,b):
    assert(num > 0)
    assert(a < b)
    angles = 2 * np.pi * np.arange(num) / num
    if a != b:
        e = (1.0 - a ** 2.0 / b ** 2.0) ** 0.5
        tot_size = sp.special.ellipeinc(2.0 * np.pi, e)
        arc_size = tot_size / num
        arcs = np.arange(num) * arc_size
        res = sp.optimize.root(
            lambda x: (sp.special.ellipeinc(x, e) - arcs), angles)
        angles = res.x
    return angles
a = 2
b = 9
n = 50

phi = angles_in_ellipse(n, a, b)
e = (1.0 - a ** 2.0 / b ** 2.0) ** 0.5
arcs = sp.special.ellipeinc(phi, e)

fig = plt.figure()
ax = fig.gca()
ax.axes.set_aspect('equal')
ax.scatter(b * np.sin(phi), a * np.cos(phi))
plt.show()
```

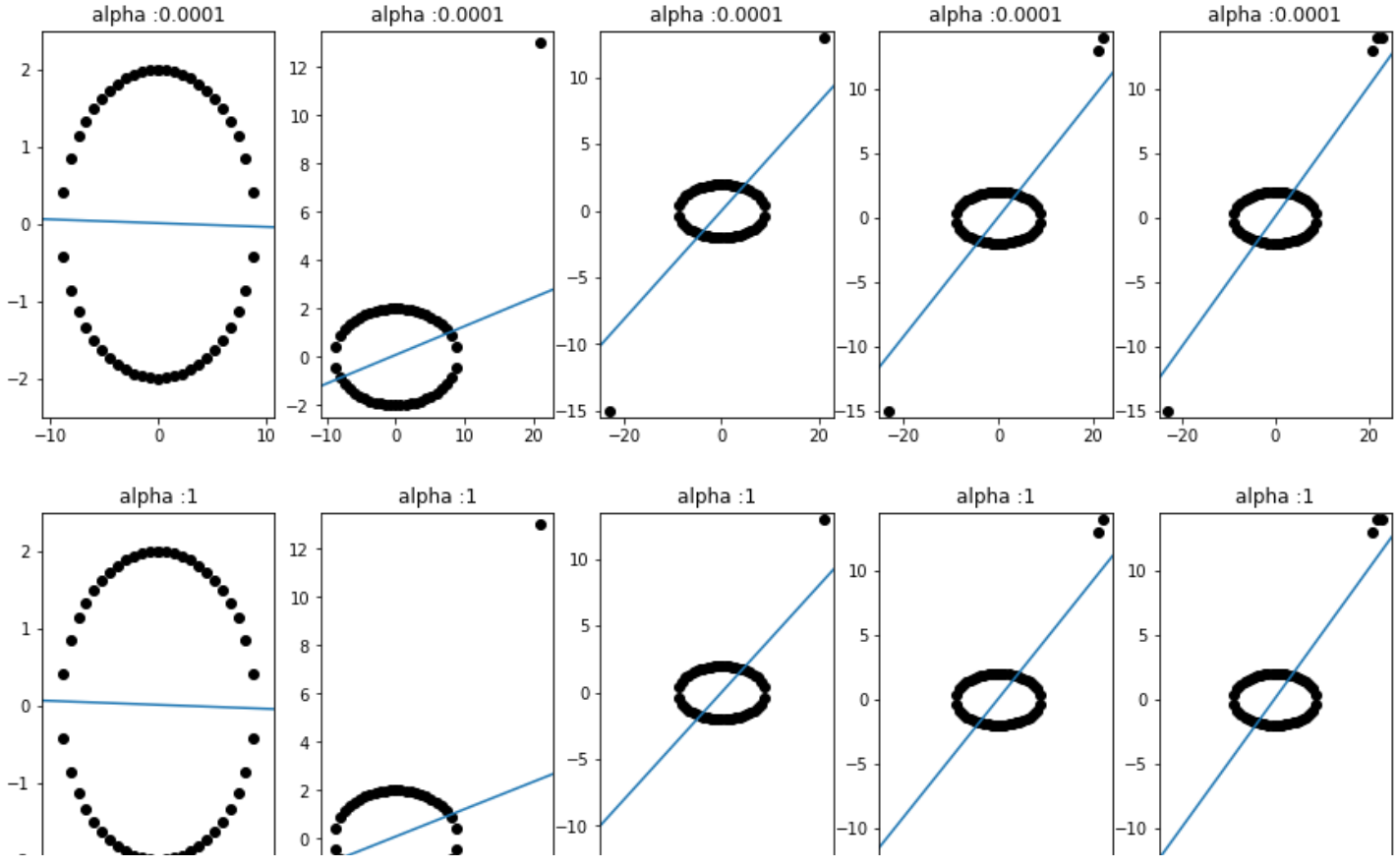


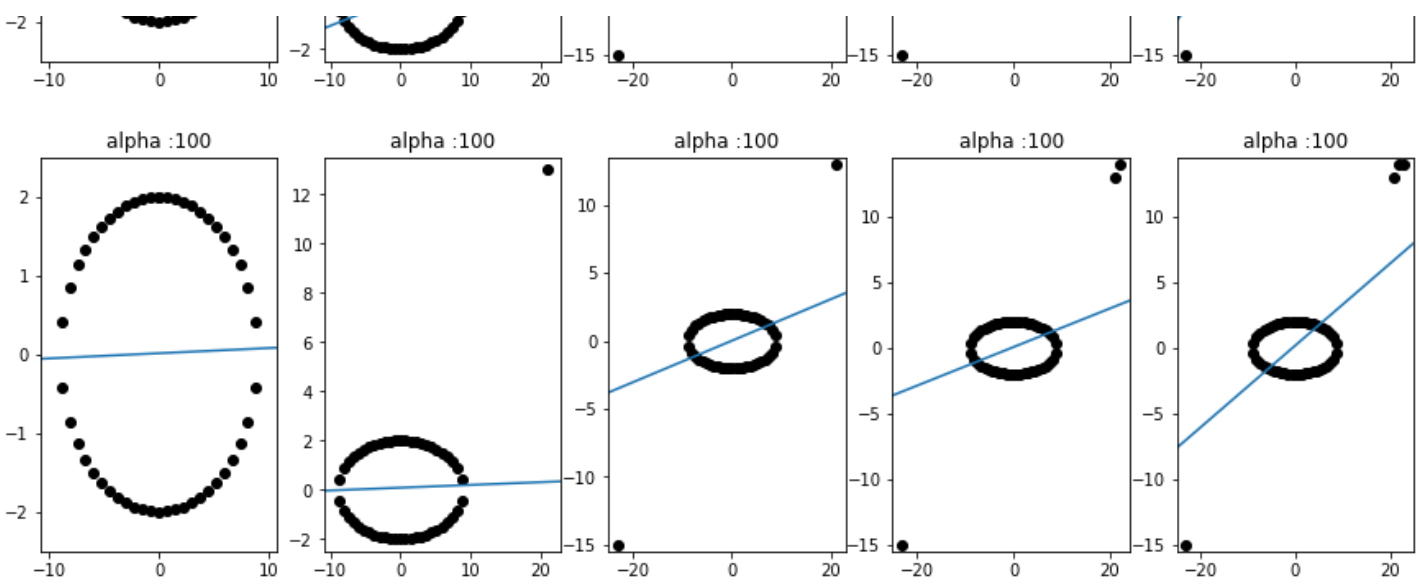
In [34]:

```
def draw_hyper_plane(coef,intercept,y_max,y_min):
    points=np.array([[((y_min - intercept)/coef), y_min],[((y_max - intercept)/coef), y_
max]])
    plt.plot(points[:,0], points[:,1])
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import SGDRegressor

alpha_lst = [0.0001,1,100]

outlier = [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]
count=1
for i in range(len(alpha_lst)):
    plt.figure(figsize = (15,15))
    k = 0
    X= b * np.sin(phi)
    Y= a * np.cos(phi)
    for j in outlier:#each outlier

        mse=0
        msx=[]
```

```
        plt.subplot(3,5,k+1)
        k+=1
        X = np.append(X,j[0]).reshape(-1,1)
        Y = np.append(Y,j[1]).reshape(-1,1)
        clf = SGDRegressor(alpha=alpha_lst[i], eta0=0.001, learning_rate='constant',rand
om_state=0)
        clf.fit(X,Y)
        y_pred=clf.predict(X[:])

        for p in range((X.shape[0])):
            mse+=(Y[p]-y_pred[p])**2/(X.shape[0])
        coef = clf.coef_
        intercept = clf.intercept_
        x_min=np.amin(Y)
        x_max=np.amax(Y)
        y_min = np.amin(X)
        y_max = np.amax(X)
        plt.title("alpha :" +str(alpha_lst[i]))
        print("MSE of plot {0}:{1}".format(count,mse))
        count+=1

        plt.xlim(y_min-2,y_max+2)
        plt.ylim(x_min-0.5,x_max+0.5)
        hyper_plane = draw_hyper_plane(coef,intercept,y_min,y_max)
        plt.scatter(X,Y,color='black')

plt.show()
```

```
MSE of plot 1:[2.62175395]
MSE of plot 2:[5.04938533]
MSE of plot 3:[7.92034582]
MSE of plot 4:[8.97524947]
MSE of plot 5:[9.56870941]
MSE of plot 6:[2.62182241]
MSE of plot 7:[5.06005229]
MSE of plot 8:[7.85299173]
MSE of plot 9:[8.91120539]
MSE of plot 10:[9.5063092]
MSE of plot 11:[2.6221512]
MSE of plot 12:[5.6671787]
MSE of plot 13:[7.45090569]
MSE of plot 14:[9.48260559]
MSE of plot 15:[8.53061503]
```





CONCLUSION : low values alpha like 0.001 and 1 tend to have more impact on hyperplane due to outlier compared to alpha with value 100