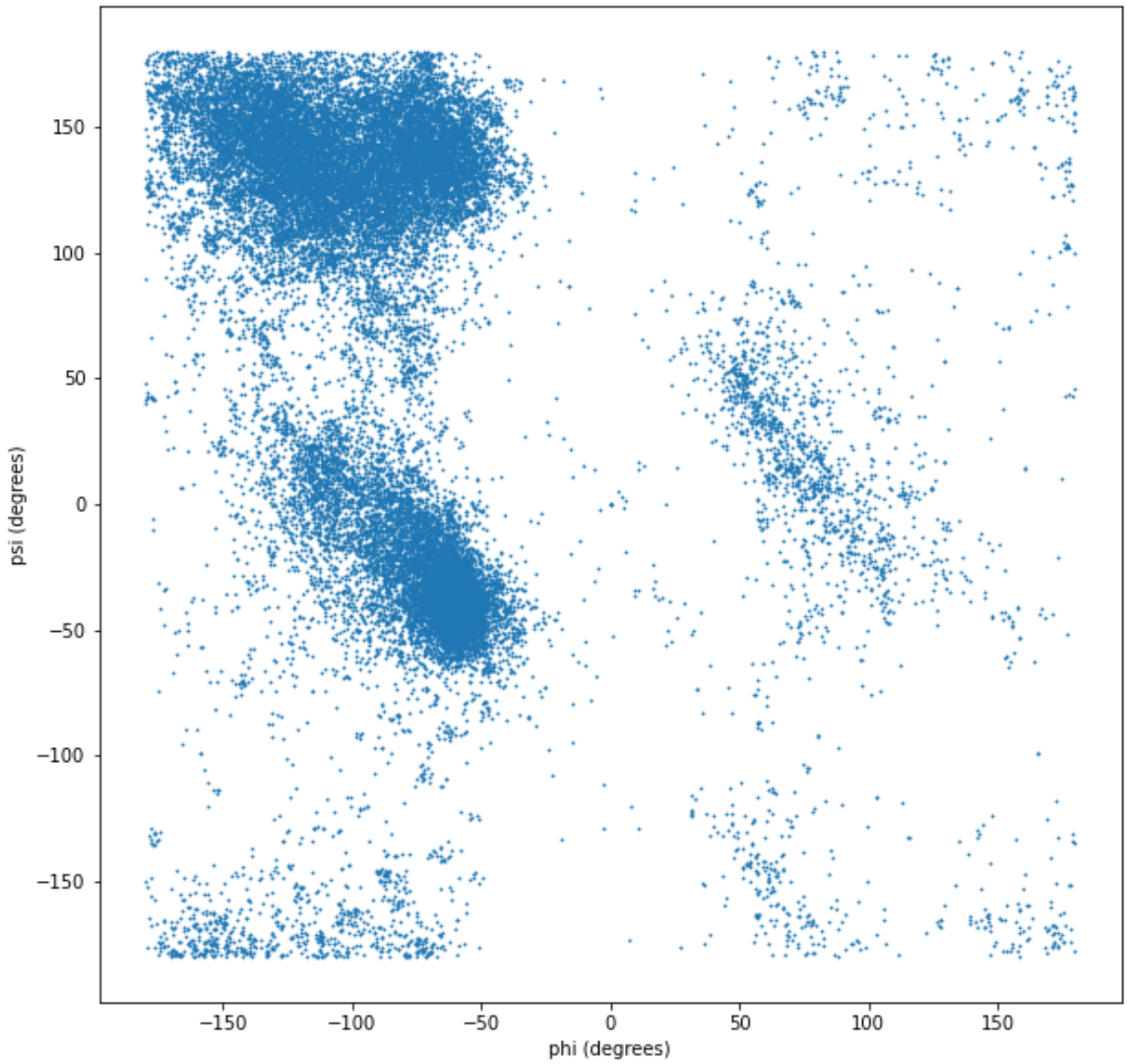
Lukas Jigberg 20 Hours

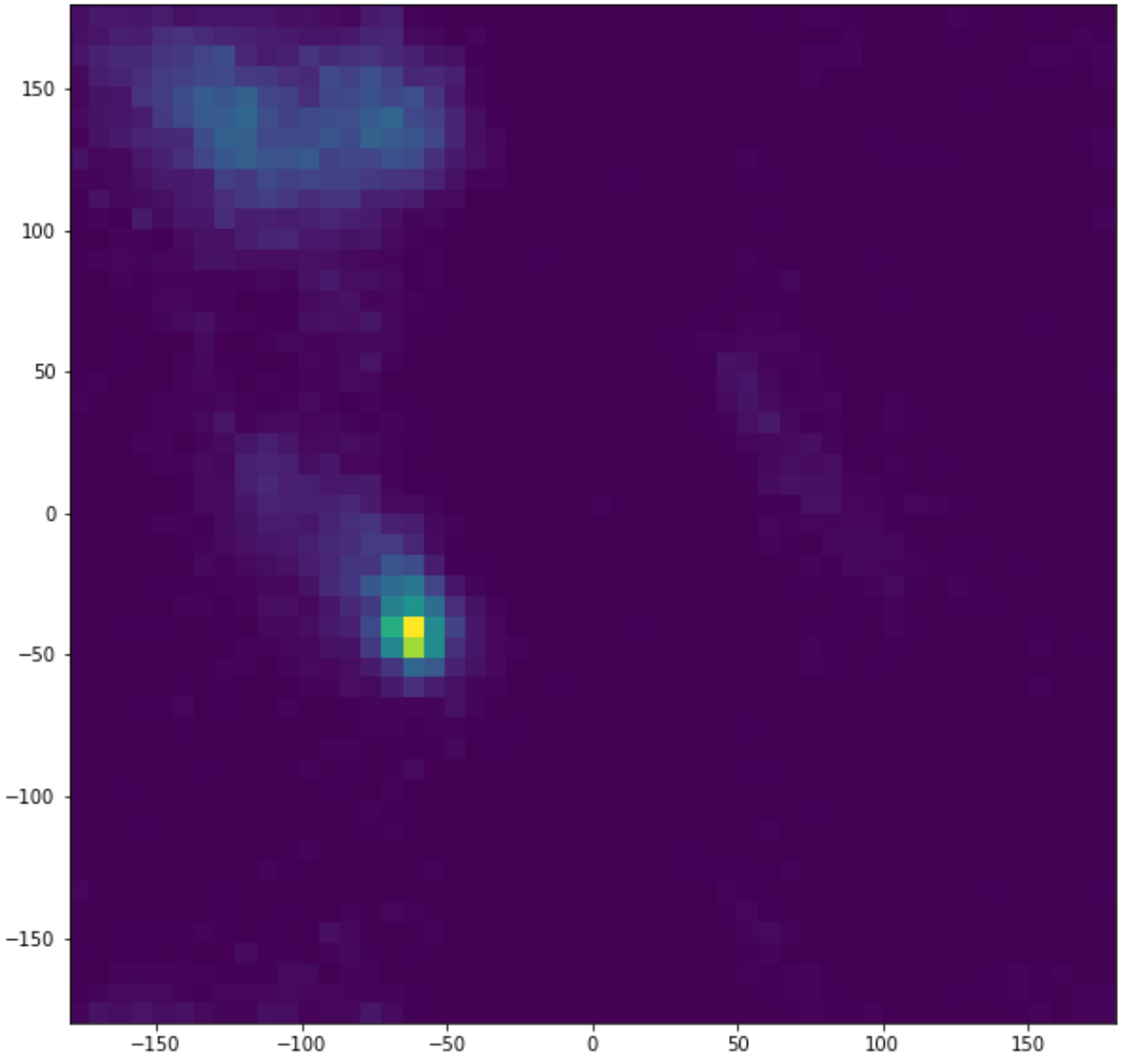
! "Run All" will take some time, grab a coffe or something meanwhile !

ScatterPlot

In [242]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import seaborn as sns

# read and remove unwanted data
data = pd.read_csv('dataSets/data_all.csv')
data = data.dropna(how='all')

#Plot
plt.figure(figsize=(10,10))
plt.scatter(x=data['phi'],y=data['psi'],s=0.75)
plt.xlabel('phi (degrees)')
plt.ylabel('psi (degrees)')
plt.show()
plt.figure(figsize=(10,10))
plt.hist2d(data['phi'],data['psi'], bins=50)
plt.show()
```


150
100
50
0
−50
−100
−150
psi (degrees)
−150
−100
−50
0
50
100
150
phi (degrees)

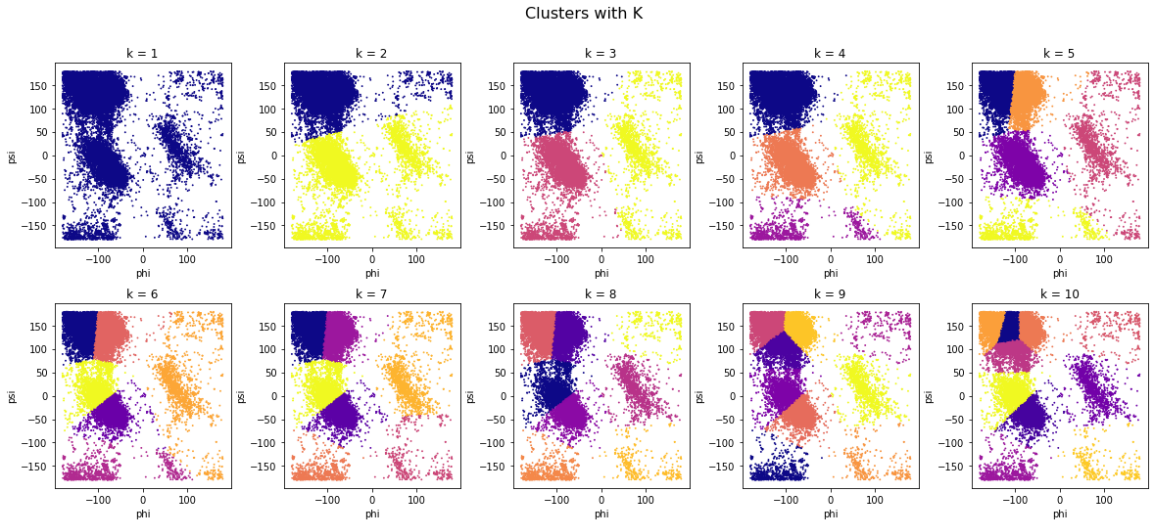

150
100
50
0
−50
−100
−150
−150
−100
−50
0
50
100
150

K Clusters

1. Experiment with K

In [243]:

```
from sklearn.cluster import KMeans

X = data.loc[:,['phi', 'psi']]
input = data.loc[:,['phi', 'psi']]
kRange = range(1,11)
fig, axs = plt.subplots(2,5, figsize=(20, 8))
wcss = []


# Loop the plot creation
i = 0
j = 0
for k in kRange:

  #Create the model
  kmean_model = KMeans(n_clusters=k, random_state=0).fit(X)

  #Predict clusters
  y_pred = kmean_model.predict(X)

  #Collect data for the elbow below
  wcss.append(kmean_model.inertia_)

  #Plot
  axs[i][j].scatter(x=X['phi'],y=X['psi'],s = 0.75, c=y_pred,cmap='plasma')
  axs[i][j].set_xlabel('phi\n')
  axs[i][j].set_ylabel('psi')
  axs[i][j].set_title(f'k = {k}')
  plt.subplots_adjust(wspace=0.3, hspace=0.3)
  j+=1
  if (j==5):
    i = 1
    j = 0
fig.suptitle('Clusters with K', fontsize=16)
plt.show()
```

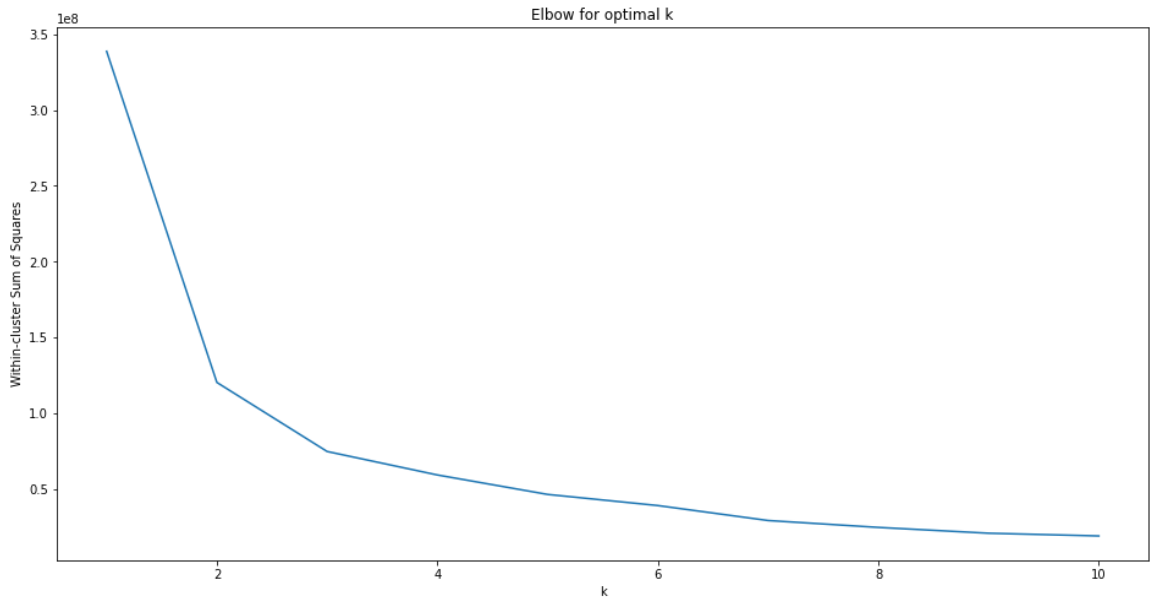


Tough some groups do look nicely clustered can we quickly see from the graphs doesn’t really match how a normal person would group the original scatter plot. I, for example, see 6 heaps of dots, that does not match the k = 6 plot we got predicted. It might be that the 'model' sees something that isn’t apparent on the scatterplot, like its depth, visible on the above heatmap. But mostly like it’s because 'k means' is being inaccurate by splitting well shaped heaps/clusters down the middle.

To be more scientific can we use the elbow method to see which of the above clusters is the best one.

In [244]:

```
#Plot with the data collected from above
plt.figure(figsize=(16,8))
plt.plot(kRange, wcss)
plt.xlabel('k')
plt.ylabel('Within-cluster Sum of Squares')
plt.title('Elbow for optimal k')
plt.show()
```



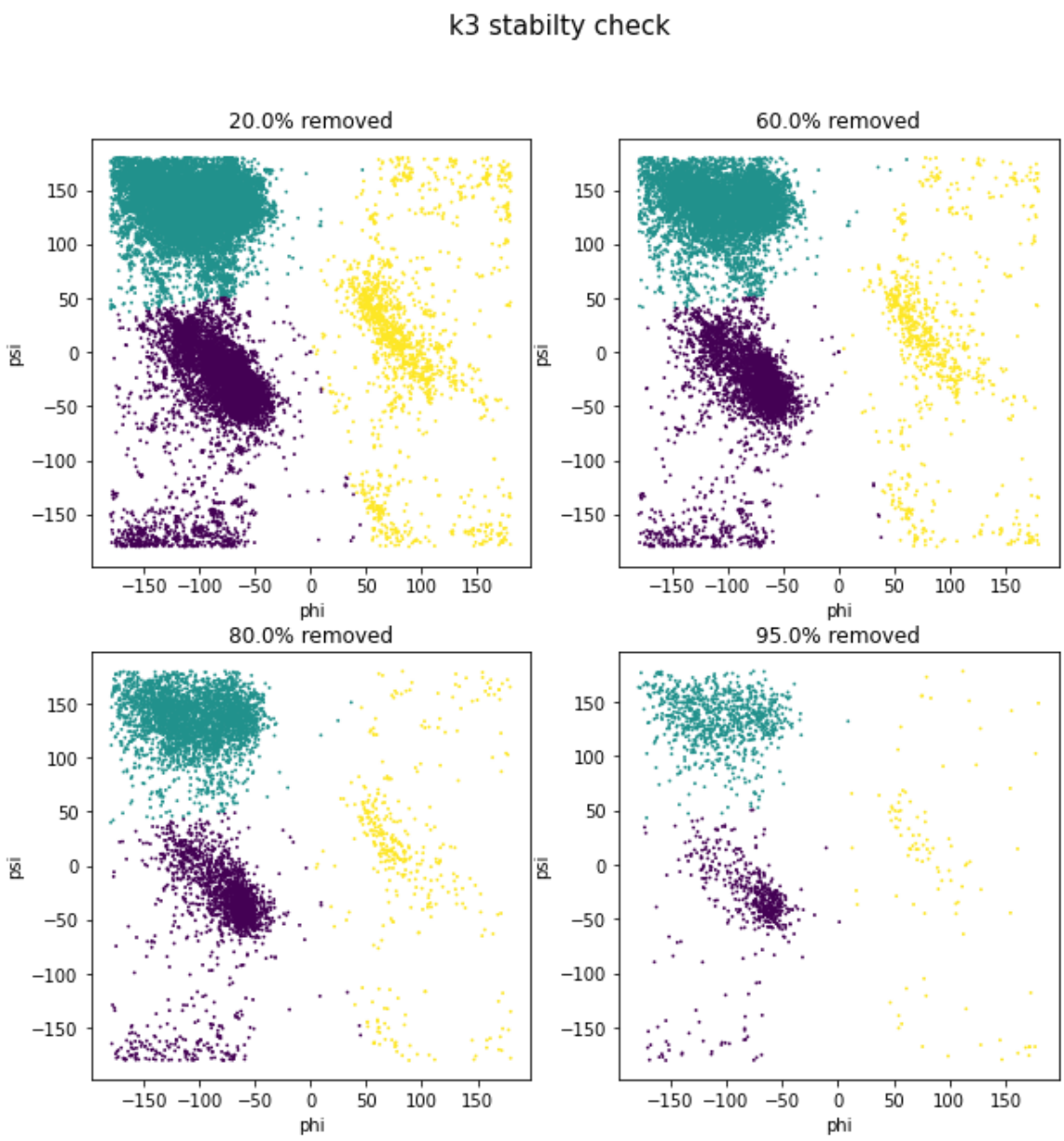
The sharpest angle is at k = 2, closely followed by k = 3. We should than logically choose one of them. Even tough 2 has the sharpest angle in the elbow graph do I personally think that the k2 plot is inaccurate. We can clearly see that there is a great divide in the middle, but the two clusters do not take this into a count. k3 however does and does not have any striking visible inaccuracy's, therefore do I choose k3 to be the appropriate value for k THIS task.

Validate k3

To get a better visualisation how 'thick'/'consistent' a cluster is can we either A, use another heat map or B, thin out the dots little by little to see where the greatest concentration is. I decided to use multiple plots to thin out the k3.

In [245]:

```
import math
np.random.seed(0)

removePercentage = [0.2, 0.6, 0.8, 0.95]
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

#Loop plot creation
i = 0
j = 0
for rem in removePercentage:

  drop_indices = np.random.choice(data.index, (math.floor(rem * data.shape[0])), replac
e=False)
  data_subset = data.drop(drop_indices)
  X, Y = data_subset['phi'], data_subset['psi']
  angles_data = np.column_stack((X,Y))

  #Our k3 model
  kmeans = KMeans(n_clusters=3, random_state=0).fit(angles_data)

  axs[i][j].set_xlabel('phi\n')
  axs[i][j].set_ylabel('psi')
  axs[i][j].set_title(f'{rem*100}% removed')
  axs[i][j].scatter(X, Y, 0.75, c=kmeans.labels_.astype(float))

  j+=1
  if (j==2):
    i = 1
    j = 0

fig.suptitle("k3 stabilty check", size=15)
plt.show()
```


k3 stabilty check
20.0% removed
60.0% removed
80.0% removed
95.0% removed
psi
phi
150
100
50
0
−50
−100
−150

When we remove dots from the plots do, we get a better visualisation of where the 'real' cluster are. On the original scatter plot there where a lot of space taken up by dots, however since we didn't see the number of dots atop each other did we not know if a space was simply a cluster or just a 'fluke'. By skimming away most dots are we left the with the spots where most dots will end up once we add more of our data, effectively confirming those spots as real clusters. \ We see for example that the hot spot on our heatmap is still a cluster on the 95% plot. We also see that the right yellow cluster and the bottom part of purple is really scattered compared to the rest.

Looking at our 80% and 95% removed plot clearly show us that there are 3 strong clusters, validating our assumption that k3 was the best k we could choose. We however also see that those 3 cluster centres (mostly purple and yellow) are really spread out, almost looking like they are encompassing clusters that should be their own clusters.

Are the clusters reasonable

For starters does they have a tendency to split groups unforgivingly, taking a group and splitting it right down the middle. To crimidine this problem would you have to manually place the clusters which isn’t ideal. \ It also forms group of dots that are very far apart creating big groups that doesn’t really blend together. Looking at the graphs above can we see that tough there are 3 clear strong clusters there are also some smaller ones that get engulfed that maybe should be their own group. Increasing K will however not adept these as expected and will duo to the problems mentioned instead create a weird cluster.

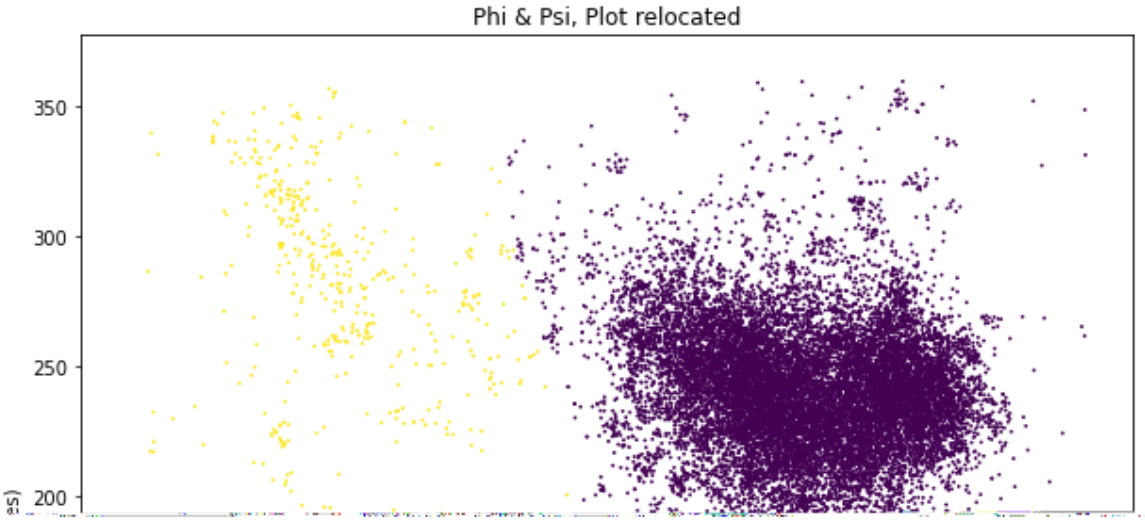
To create a better-looking cluster would we either have to improve on the K method or use another method to create a cluster. This new cluster would hopefully not generate the same problems as K-means and would create a better-looking cluster.

Data Change

An important factor for this assignment’s plots are that they essentially loop around. 360 degrees + 1 will not be outside our graph but will show up x = 1. This means that the opposite ends of the graph is in reality right next to each other. Which can be seen as the bottom of the original scatterplot suspiciously fits onto the top. \ Due to how the molecule works can we 'move' the data points, as long as we keep their degrees relation allowing us to place the scatterplot in a more favourable position. We can see two 'gaps/lines' with very few dots where psi = -100 and phi = 0. The best option would be to place the intersection of those gaps at the low left corner. Making it so that no group/cluster is split in half by our plot.

In [246]:

```
#The data shift
phi_angle_offset = 360
psi_angle_offset = 100

data['phi shifted 360°'] = data.apply(lambda row: ((row.phi + phi_angle_offset) % 360),
axis=1)
data['psi shifted 100°'] = data.apply(lambda row: ((row.psi + psi_angle_offset) % 360),
axis=1)

X = data[['phi shifted 360°', 'psi shifted 100°']]
X = StandardScaler().fit_transform(X)

#Kmeans model % predict
kmean_model = KMeans(n_clusters=3, random_state=0).fit(X)
y_predK = kmean_model.predict(X)

#Plot
plt.figure(figsize=(10,10))
plt.scatter(x=data['phi shifted 360°'], y=data['psi shifted 100°'], c=y_predK, s = 0.75
)
plt.xlabel('phi (degrees)')
plt.ylabel('psi (degrees)')
plt.title('Phi & Psi, Plot relocated')
plt.show()
```



Instantly can we see an improvement. No group is cut by the border, there is only a small number of dots around the edges and we instantly see the difference of what connects dots/group together more easily now.

DBSCAN Cluster

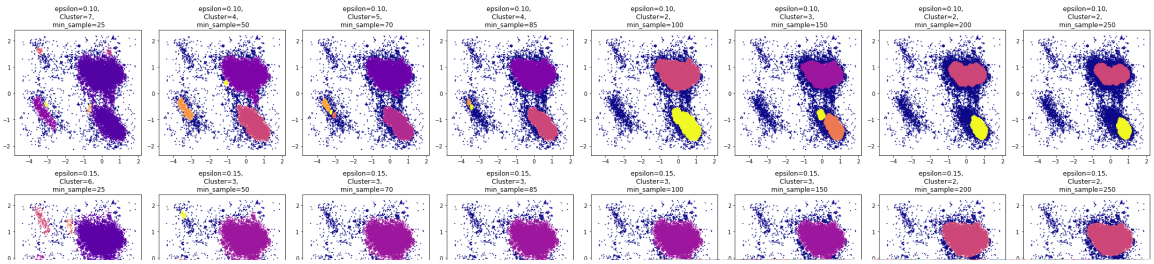
Motivate DBSCAN values

We need to decide epsilon (eps) and the minimum number for the DBSCAN. A quick and good solution is to plot DCSCAN's with different value and picking the one/ones that show the best result.

!Note that henceforth will we use the updated data plot from the previous exercise.

In [247]:

```
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors

X = data[['phi shifted 360°', 'psi shifted 100°']]

# eps & minSamples
min_samples = [25,50,70,85,100,150,200,250]
eps = [0.1,0.15,0.2,0.3,0.4,0.5,0.6]

X = StandardScaler().fit_transform(X)
fig, axs = plt.subplots(7,8, figsize=(30,30))


#Loop every DBSCAN subplot creation
for i in range(0, len(eps)):
  for j in range (0, len(min_samples)):

    print(i,j)

    # Create DBSCAN with our different test data
    dbScan = DBSCAN(eps=eps[i], min_samples=min_samples[j]).fit(X)
    labels = dbScan.labels_

    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
    n_noise_ = list(labels).count(-1)

    y_pred = dbScan.fit_predict(X)

    #Plot
    axs[i][j].scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.9)
    axs[i][j].set_title('epsilon=%.2f, \nCluster=%d, \n min_sample=%d' %(eps[i], n_clus
ters_, min_samples[j]))

plt.tight_layout()
plt.show()
```


epsilon=0.10,
Cluster=7,
min_sample=25
epsilon=0.10,
Cluster=4,
min_sample=50
epsilon=0.10,
Cluster=5,
min_sample=70
epsilon=0.10,
Cluster=4,
min_sample=85
epsilon=0.10,
Cluster=2,
min_sample=100
epsilon=0.10,
Cluster=3,
min_sample=150
epsilon=0.10,
Cluster=2,
min_sample=200
epsilon=0.10,
Cluster=2,
min_sample=250
epsilon=0.15,
Cluster=6,
min_sample=25
epsilon=0.15,
Cluster=3,
min_sample=50
epsilon=0.15,
Cluster=3,
min_sample=70
epsilon=0.15,
Cluster=3,
min_sample=85
epsilon=0.15,
Cluster=3,
min_sample=100
epsilon=0.15,
Cluster=3,
min_sample=150
epsilon=0.15,
Cluster=2,
min_sample=200
epsilon=0.15,
Cluster=2,
min_sample=250

This is a half decent way to pick a eps and min. Firstly would we really want a real professional in the protein molecule field to determine which cluster plot is the most viable, not picking one ourselves. Secondly is it very likely that we miss out on a slightly better solution. Since we can't check all possibilities. However, considering the exercise do we just need 2 workable values that give us a decent plot that matches our expectations. \ Due to the nature of DBSCAN does it look like we must compromise on many plots on whatever we 2 clusters on the right or if we want the chunk on the left to be a cluster at all.

Eps 0.15 and min (70, 85, 100) do look very promising. At first glance does it look like they are not stretched all the way out, but that can just be edge noice that isn't really 'connected'. I make another subplot this time with values closer to the ones we spotted to try to find even better values.

In [248]:

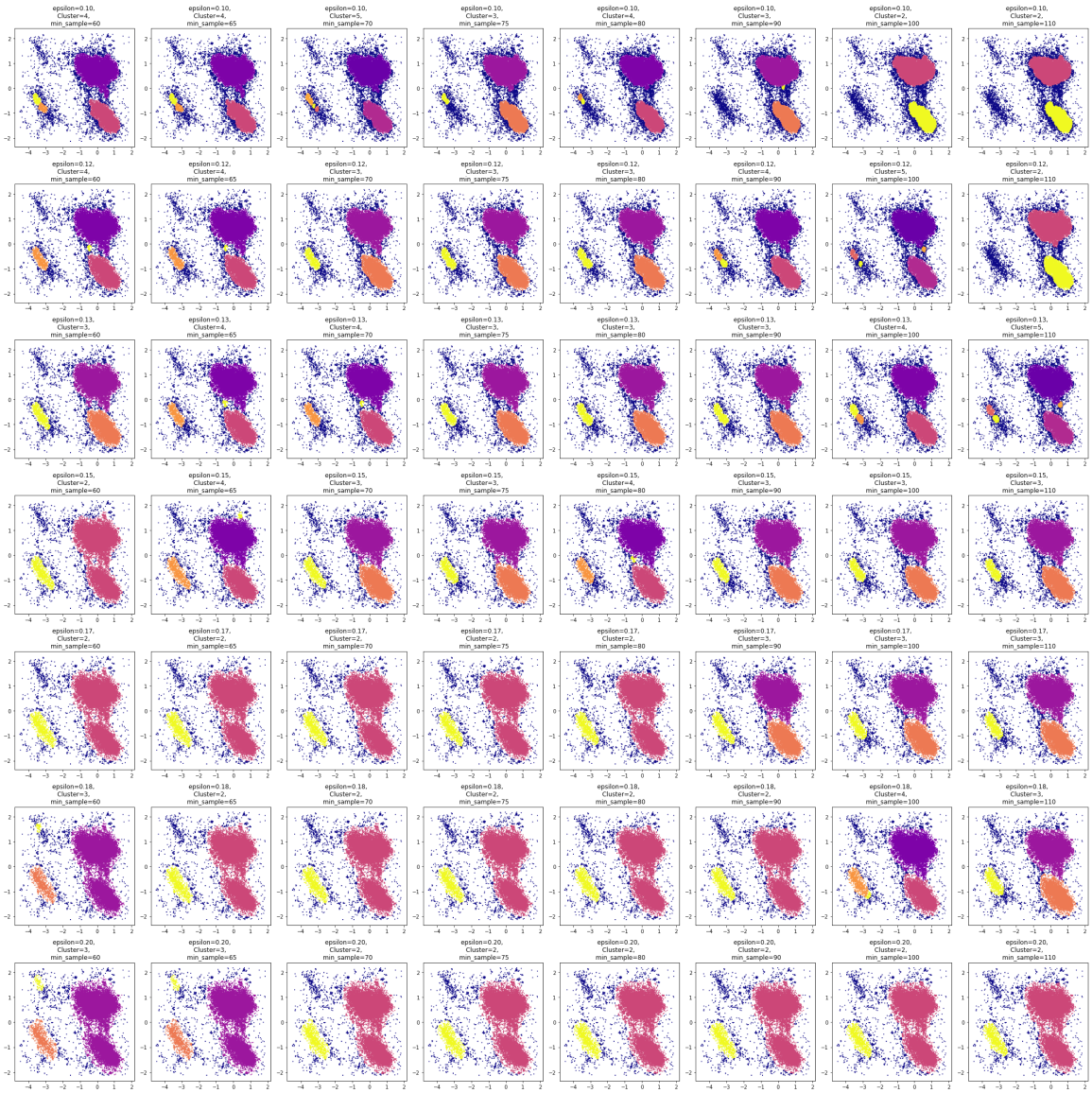
```
X = data[['phi shifted 360°', 'psi shifted 100°']]

# eps & minSamples
min_samples = [60,65,70,75,80,90,100,110]
eps = [0.10,0.12,0.13,0.15,0.17,0.18,0.2]
length_samples = len(min_samples)
length_eps = len(eps)
n_samples = 1000

fig, axs = plt.subplots(7,8, figsize=(30,30))
X = StandardScaler().fit_transform(X)

#Loop every DBSCAN subplot creation
for i in range(0, length_eps):
  for j in range (0, length_samples):

    print(i,j)

    # Create DBSCAN with our different test data
    dbScan = DBSCAN(eps=eps[i], min_samples=min_samples[j]).fit(X)
    labels = dbScan.labels_

    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
    n_noise_ = list(labels).count(-1)

    y_pred = dbScan.fit_predict(X)

    #Plot
    axs[i][j].scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.9)
    axs[i][j].set_title('epsilon=%.2f, \nCluster=%d, \n min_sample=%d' %(eps[i], n_clus
ters_, min_samples[j]))

plt.tight_layout()
plt.show()
```

This is a half decent way to pick a eps and min. Firstly would we really want a real professional in the protein molecule field to determine which cluster plot is the most viable, not picking one ourselves. Secondly is it very likely that we miss out on a slightly better solution. Since we can't check all possibilities. However, considering the exercise do we just need 2 workable values that give us a decent plot that matches our expectations. \ Due to the nature of DBSCAN does it look like we must compromise on many plots on whatever we 2 clusters on the right or if we want the chunk on the left to be a cluster at all.

Eps 0.15 and min (70, 85, 100) do look very promising. At first glance does it look like they are not stretched all the way out, but that can just be edge noice that isn’t really 'connected'. I make another subplot this time with values closer to the ones we spotted to try to find even better values.

In [248]:

```
X = data[['phi shifted 360°', 'psi shifted 100°']]

# eps & minSamples
min_samples = [60,65,70,75,80,90,100,110]
eps = [0.10,0.12,0.13,0.15,0.17,0.18,0.2]
length_samples = len(min_samples)
length_eps = len(eps)
n_samples = 1000

fig, axs = plt.subplots(7,8, figsize=(30,30))
X = StandardScaler().fit_transform(X)

#Loop every DBSCAN subplot creation
for i in range(0, length_eps):
  for j in range (0, length_samples):

    print(i,j)

    # Create DBSCAN with our different test data
    dbScan = DBSCAN(eps=eps[i], min_samples=min_samples[j]).fit(X)
    labels = dbScan.labels_

    # Number of clusters in labels, ignoring noise if present.
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
    n_noise_ = list(labels).count(-1)

    y_pred = dbScan.fit_predict(X)

    #Plot
    axs[i][j].scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.9)
    axs[i][j].set_title('epsilon=%.2f, \nCluster=%d, \n min_sample=%d' %(eps[i], n_clus
ters_, min_samples[j]))

plt.tight_layout()
plt.show()
```



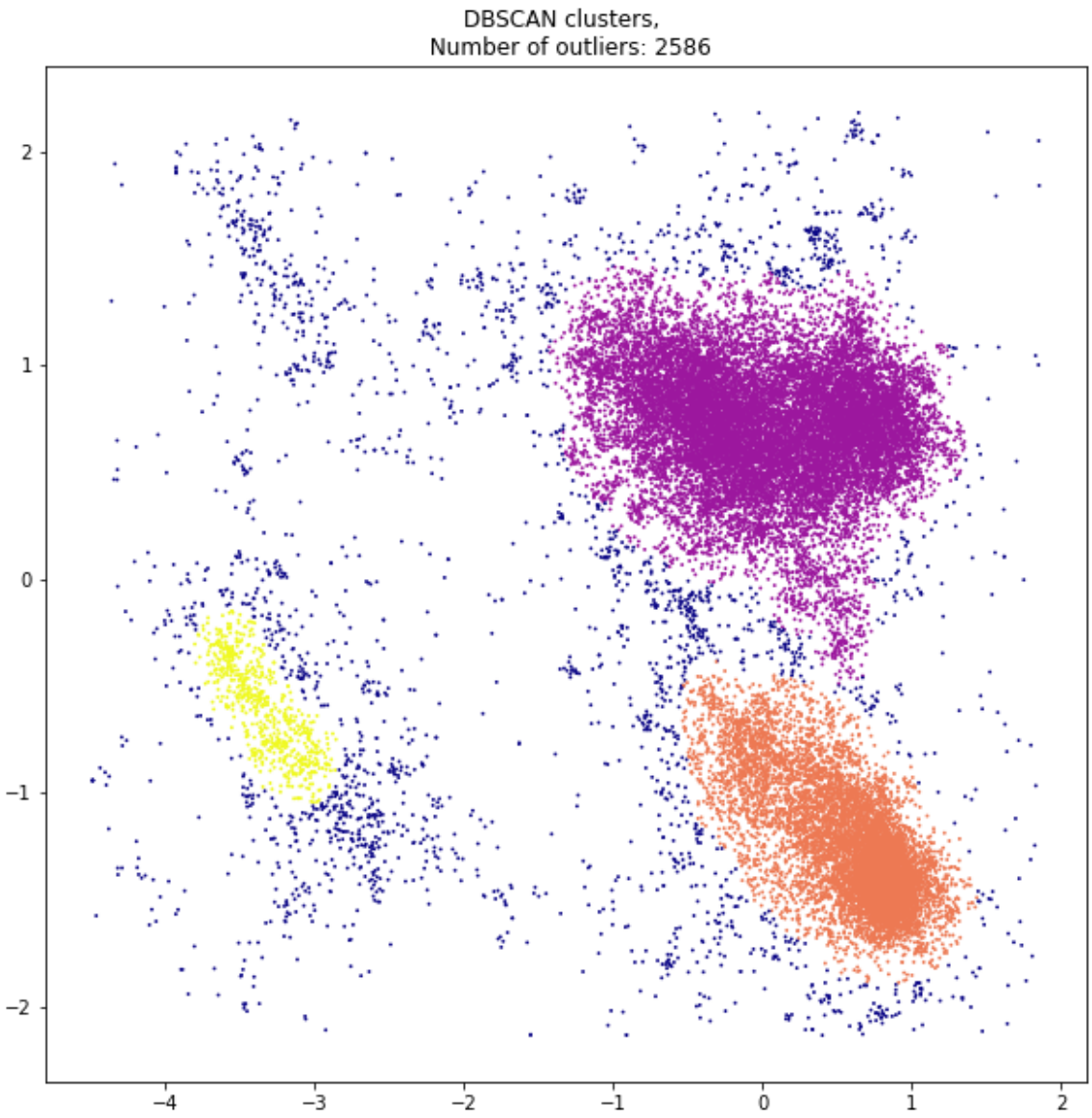
Again, an expert would be able to choose the best one, I however will choose eps = 0.13, min = 75. It looks the best in my personal opinion. It keeps the 2 clusters on the right separate, doesn't pick up to much noice dots and crates a cluster on the left.

In [249]:

```
X = data[['phi shifted 360°', 'psi shifted 100°']]
X = StandardScaler().fit_transform(X)

# Create DBSCAN model and remove noise
model_dbscan = DBSCAN(eps=0.13, min_samples=75).fit(X)
labels = model_dbscan.labels_

# Number of clusters in labels, ignoring noise if present.
number_of_outliers = list(labels).count(-1) # outliers are given the label -1.
y_pred = model_dbscan.fit_predict(X)

# Plot
plt.scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.75)
plt.title(f'DBSCAN clusters, \n Number of outliers: {number_of_outliers}')
```

Out[249]:

```
Text(0.5, 1.0, 'DBSCAN clusters, \n Number of outliers: 2586')
```

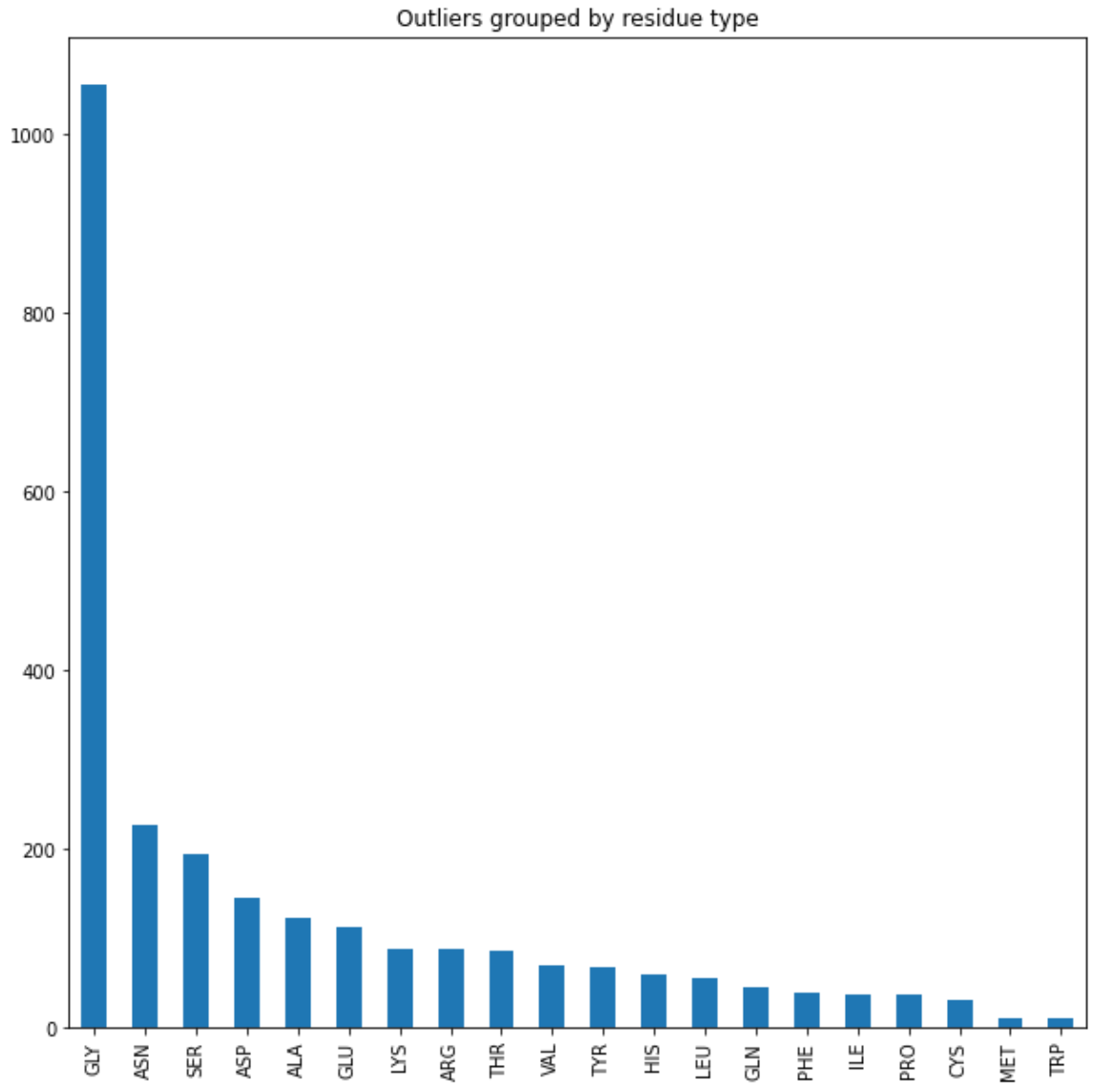


eps = 0.13 \ min_val = 75

Highlight outliers with bars

In [250]:

```
data['labels'] = list(labels)

#Create a barGraph with the saved labels data
data_outliers = data[(data['labels'] == -1)]
bar = data_outliers['residue name'].value_counts(sort=True).plot.bar()
plt.title('Outliers grouped by residue type')
plt.show()
```



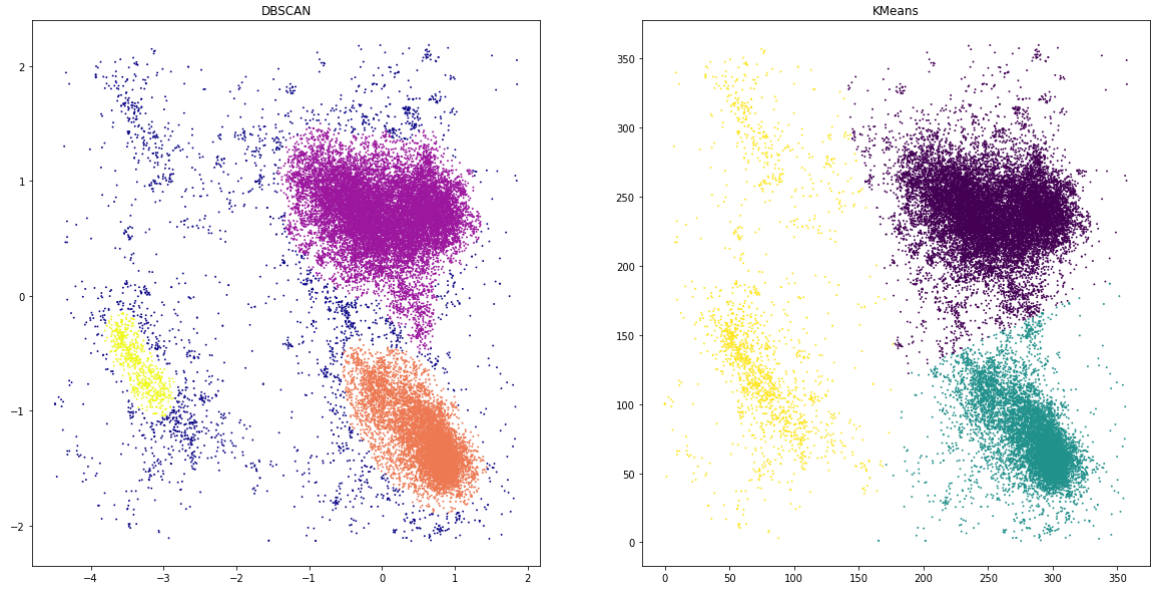
Compare K-means vs DBSCAN

In [251]:

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

#Create DBSCAN
ax1.scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.75)
ax1.set_title('DBSCAN')

#Create KMeans
ax2.scatter(x=data['phi shifted 360°'], y=data['psi shifted 100°'], c=y_predK, s = 0.75
)
ax2.set_title('KMeans')
```

Out[251]:

```
Text(0.5, 1.0, 'KMeans')
```



Let start with the obvious problems discussed with KMeans.

1. The clusters uses every dot even if its close by.
2. The K had to be manually chosen to get the best score.
3. The clusters get really big unless cut off by another cluster.
4. It does tend to split clusters in half (showed on previous plots)

Now for the problems with DBSCAN

1. It requires even more finessing to get a good score than K-means

Clearly does DBSCAN deliver a better result. It has the ability to disregard dots as noice and can find 'boundaries' where a cluster should end. This is probably the most important feature of it, if we were to show the KMeans data would it be apparent instantly that it can't be true. Yellow takes up the whole left side and increasing to k to 4 would only destroy the right top cluster instead of creating a new one. \ DBSCAN does require a bit more technic to get a good result, but the upgrade is well worth it. Going back to the lecture about clusters and grouping, most humans would instantly choose the DBSCAN plot out of the two as it looks a lot more believable. The blue noice might look weird in some places but clustering it would result in problems someplace else and in the end having some noice is a lot better than having none.

Is DBSCAN Susceptible to change

That depends on what a small change is, as seen on the many subplots of epsilon and minimum number do we get a small change when we increase/decrease both. There is noticeable change when we alter eps between 0.1-0.2 and min between 60 - 110 , some cluster get bigger and some disappear or are created. \
It’s of course the plot that will decide which of the two has the most effect, but it’s the results of both that decide the plot. Our plot is both thick and compact on some places but is also scattered on others. Take for example the hot spot we'd seen earlier and the cluster on the left. It is clear that there is some form of group building on the left but compared to the massive compact hotspot might it be regarded as noise. When trying to find eps and min did the problem come out of having to compromise. Trying to create a cluster on the left did alter the clusters on the right as the noise surrounding them got absorbed into them. Just changing the values a little would in sense (again it depends on what a small change is) alter the plot. \ To get a better understanding of what is an acceptable change for the plot would be by contacting an expert on the subject, that can tell how much noise is truly noise and what should be in the clusters.

Amino Acids

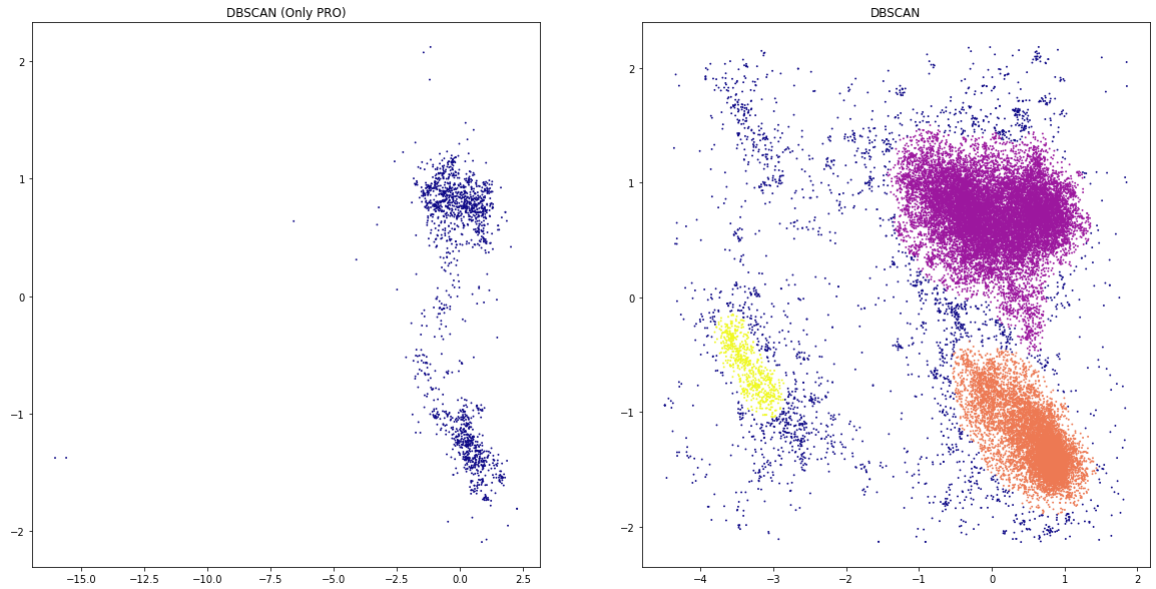
Pro

In [252]:

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

#Filter PRO
df_PRO = data[(data['residue name'] == 'PRO')]
X = df_PRO[['phi shifted 360°', 'psi shifted 100°']]
X = StandardScaler().fit_transform(X)

# create DBSCAN Method with the same eps and minSample
model_dbscan = DBSCAN(eps=0.13, min_samples=75).fit(X)
labels = model_dbscan.labels_
number_of_outliers = list(labels).count(-1) # outliers are given the label -1.
y_predPro = model_dbscan.fit_predict(X)

# Plot
ax1.scatter(X[:,0], X[:,1],c=y_predPro, cmap='plasma', s = 0.75)
ax1.set_title("DBSCAN (Only PRO)")

X = data[['phi shifted 360°', 'psi shifted 100°']]
X = StandardScaler().fit_transform(X)

#Plot Kmeans
ax2.scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.75)
ax2.set_title('DBSCAN')
```

Out[252]:

```
Text(0.5, 1.0, 'DBSCAN')
```



Having the graph right next to each other makes it clear that PRO fits pretty nicely into the two largest most concentrated clusters. This is also backed up by our bar graph that shows that PRO was one of those who contributed the least to the noise in the graph.

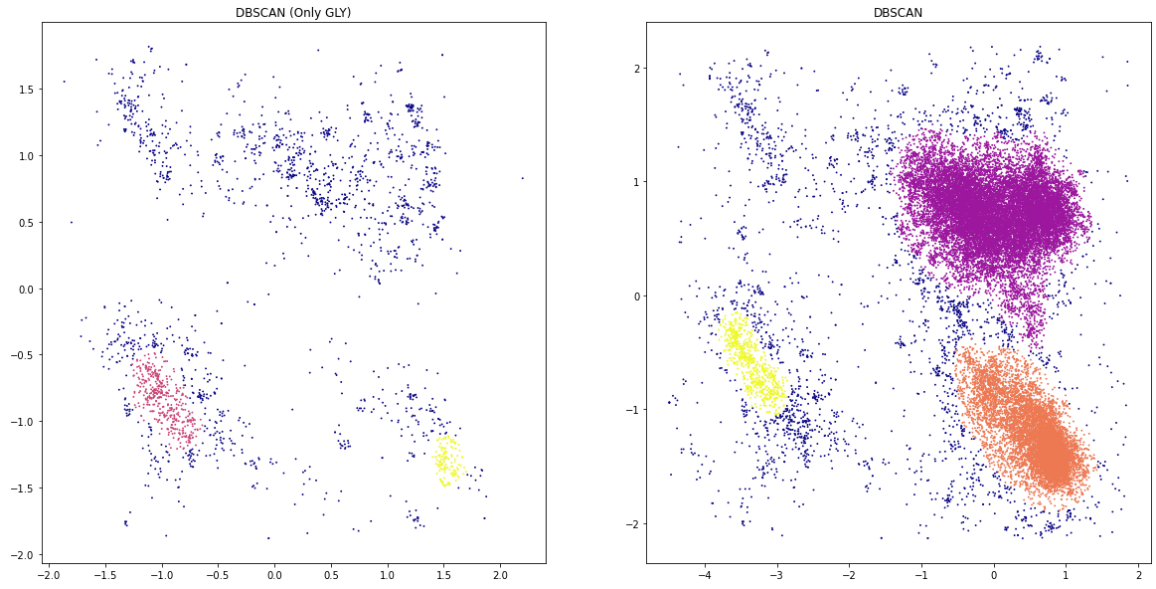
GLY

In [253]:

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

#Filter GLY
df_GLY = data[(data['residue name'] == 'GLY')]
X = df_GLY[['phi shifted 360°', 'psi shifted 100°']]
X = StandardScaler().fit_transform(X)

# create DBSCAN Method with the same eps and minSample
model_dbscan = DBSCAN(eps=0.13, min_samples=75).fit(X)
labels = model_dbscan.labels_
number_of_outliers = list(labels).count(-1) # outliers are given the label -1.
y_predGLY = model_dbscan.fit_predict(X)

# Plot
ax1.scatter(X[:,0], X[:,1],c=y_predGLY, cmap='plasma', s = 0.75)
ax1.set_title("DBSCAN (Only GLY)")

X = data[['phi shifted 360°', 'psi shifted 100°']]
X = StandardScaler().fit_transform(X)

#Plot
ax2.scatter(X[:,0], X[:,1],c=y_pred, cmap='plasma', s=0.75)
ax2.set_title('DBSCAN')
```

Out[253]:

```
Text(0.5, 1.0, 'DBSCAN')
```



GLY is very scattered and have a hard time to even form a enough concentrated cluster. We can see, also saw from the bar graph, that GLU contributes the most to the noise of the graph. Some dots do overlap with the clusters on the right, but most are situated in between or just close by the clusters. GLY does however contribute the most to the yellow cluster on the left.