## Importing the libraries

In [278]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

**Importing Data_set = Iris**

In [294]:
```
iris = pd.read_csv('D:\ML\iris.csv')
iris.head()
```

Out[294]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

In [280]:
```
iris.describe() #You can see below the descriptive statistics of numeri
cal variables in the output such as total count, mean,
#standard deviation, minimum and maximum values and three quantiles of
 the data (25%,50%,75%).
```

Out[280]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |

|  | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| **mean** | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 38.250000 | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 75.500000 | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 112.750000 | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 150.000000 | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

In [281]:
```
iris.drop(['Id'], axis = 1, inplace = True) #Dropping one columns(Id) a
s it's not important for our model
iris.head(10)
```

Out[281]:

|  | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **5** | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| **6** | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| **7** | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **8** | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| **9** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |

In [282]:
```
iris.columns = ['SL', 'SW', 'PL', 'PW','Species']
#Here I've changed the headers for ease: Here we did not check the inde
xes so writing all parameters or labels
```

In [283]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   SL       150 non-null    float64
 1   SW       150 non-null    float64
 2   PL       150 non-null    float64
 3   PW       150 non-null    float64
 4   Species  150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.3+ KB
```

In [284]:
```
iris
```

Out[284]:

| | SL | SW | PL | PW | Species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

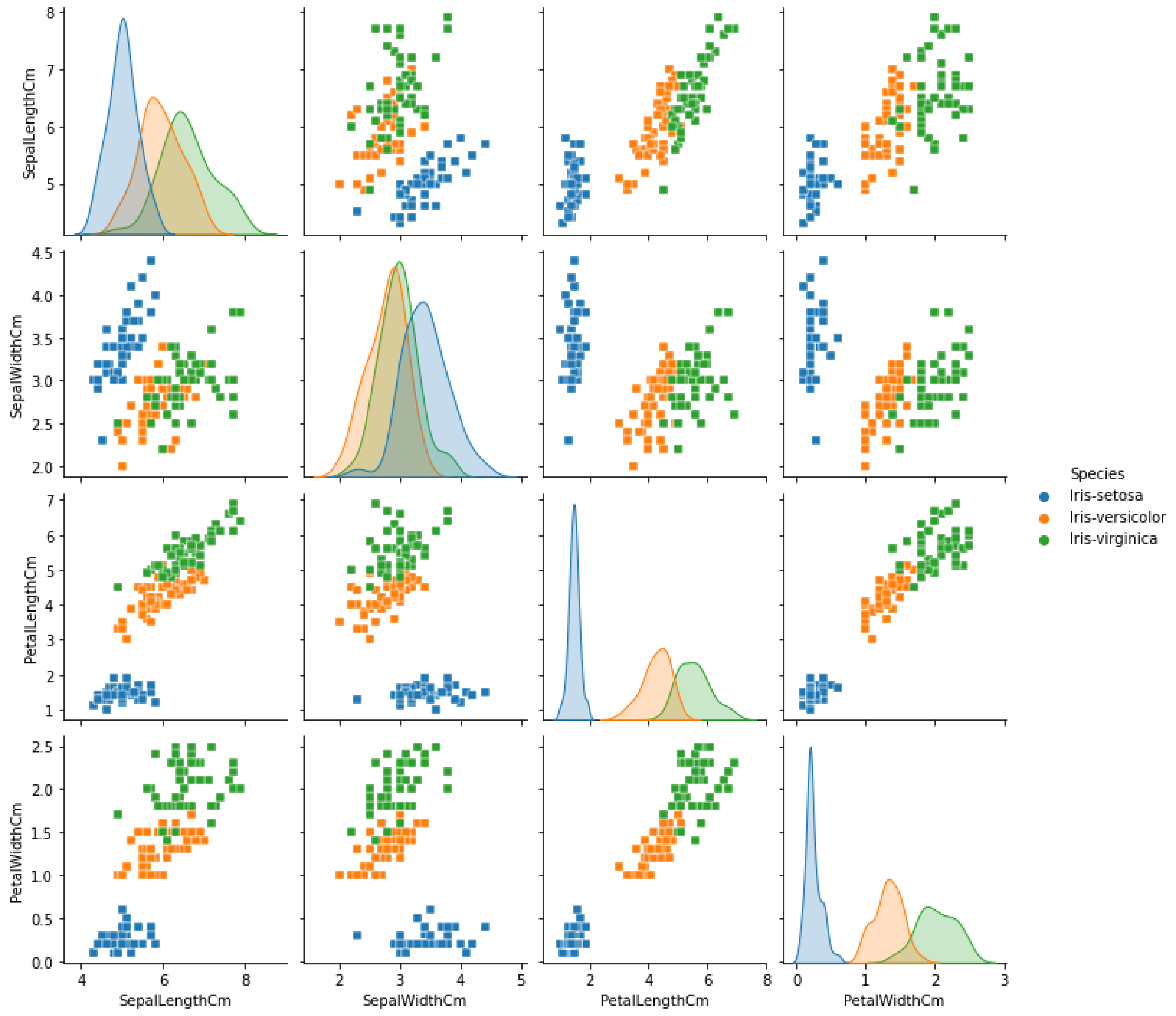
150 rows × 5 columns

## Exploratory Data Analysis

#### Visualization

In [285]:
```
sns.pairplot(tmp, hue='Species', markers='s')
#the hue parameter determines which column in the dataFrame should be u
sed for colour encoding.
```

Out[285]:
```
<seaborn.axisgrid.PairGrid at 0x25961f50>
```



We see that iris-setosa(IN BLUE) is easily separable from the other two. Especially when we can see in different colors for corresponding Labels like above. But our mission was finding the Labels that we didn't knew at all, So Let's create a suitable scenario.

```
In [286]: iris.isnull().sum()#there are no null values
```

```
Out[286]:
```

```
SL         0
SW         0
PL         0
PW         0
Species    0
dtype: int64
```

## Adjusting the Dataset for Unsupervised Learning

we will not use labels column in "new" Dataset

In [287]:
```
features = df.loc[:,["SL","SW","PL","PW"]]
features
```

Out[287]:

|  | SL | SW | PL | PW |
|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |
| **...** | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 |

150 rows × 4 columns

Now we dont know the real amount of labels

# Implementing the K-means Clustering

## An introduction with KMeans Cluster analysis

#### In K-Means the K is nothing but the no of clusters, and it follows a few matrics like: 1)-Eucledian Ditance and 2)-Manhattan distance.

#### I've used Elbow method to find the amount of clusters. Elbow method gives us an idea on what a good k number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids.

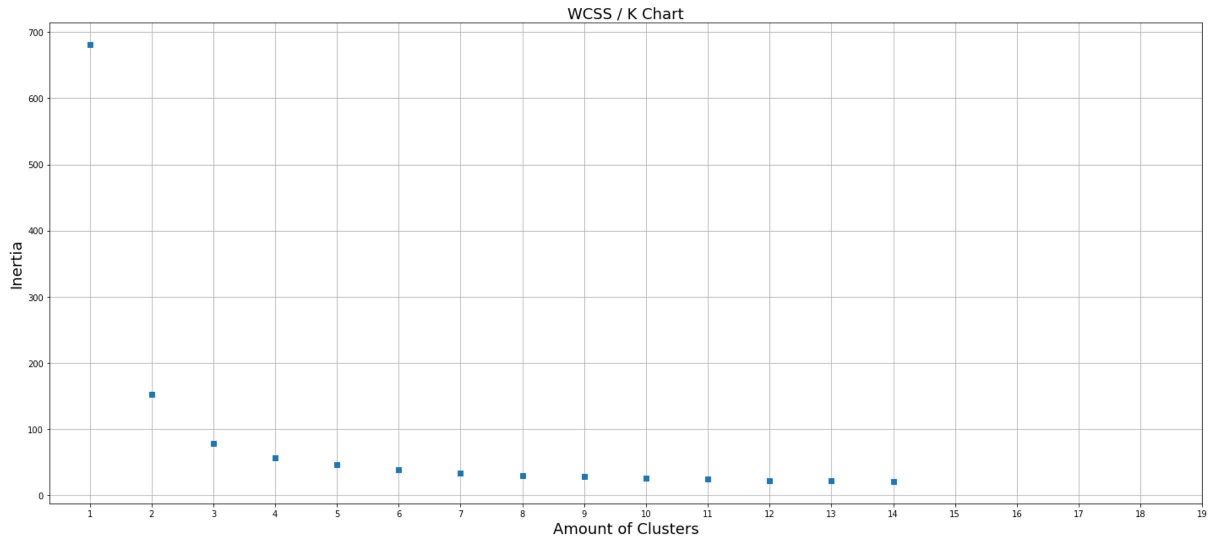
In [288]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=90) #here the Question is WHY?? 90 the answe
r is because we don't know the right amount of labels
```

#### Les's finding the correct amout of the labeled clusters by visualization

for that we'll use the 'ELBOW RULE', which is basically looking for a plot line that respectively has a slope nearest to 90 degrees compared to y axis and be smallest possible. (yes, looks like an elbow )

In [289]:
```
from sklearn.cluster import KMeans
wcss=[] # means = within cluster-sum of squared
```

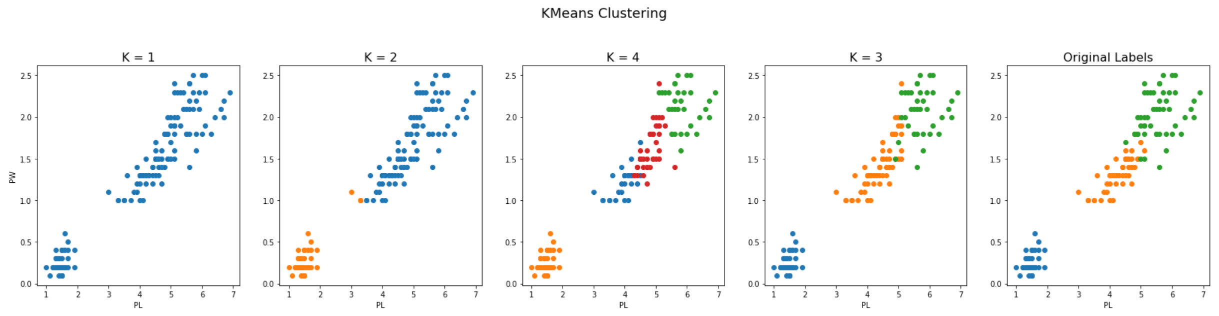
In [290]:
```
for k in range(1,15):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(features)
    wcss.append(kmeans.inertia_)


plt.figure(figsize=(20,9))
plt.title("WCSS / K Chart", fontsize=18)
plt.plot(range(1,15),wcss,"s")
plt.grid(True)
plt.xlabel("Amount of Clusters",fontsize=18)
plt.ylabel("Inertia",fontsize=18)
plt.xticks(range(1,20))
plt.tight_layout()
plt.show()
```



In the above visualization we can see an Elbow which showing the 3 Clusters.

### We will double the check the amount of clusters in our dataset

In [291]:
```
plt.figure(figsize=(28,6))

plt.suptitle("KMeans Clustering",fontsize=18)


plt.subplot(1,5,1)
plt.title("K = 1",fontsize=16)
plt.xlabel("PL")
plt.ylabel("PW")
plt.scatter(features.PL,features.PW)


plt.subplot(1,5,2)
plt.title("K = 2",fontsize=16)
plt.xlabel("PL")
kmeans = KMeans(n_clusters=2)
features["labels"] = kmeans.fit_predict(features)
plt.scatter(features.PL[features.labels == 0],features.PW[features.labe
ls == 0])
plt.scatter(features.PL[features.labels == 1],features.PW[features.labe
ls == 1])

# dropping the labels as we want to use only the features.
features.drop(["labels"],axis=1,inplace=True)

plt.subplot(1,5,4)
plt.title("K = 3",fontsize=16)
plt.xlabel("PL")
kmeans = KMeans(n_clusters=3)
features["labels"] = kmeans.fit_predict(features)
plt.scatter(features.PL[features.labels == 0],features.PW[features.labe
ls == 0])
plt.scatter(features.PL[features.labels == 1],features.PW[features.labe
ls == 1])
plt.scatter(features.PL[features.labels == 2],features.PW[features.labe
ls == 2])

# dropping the lables and using the features
features.drop(["labels"],axis=1,inplace=True)

plt.subplot(1,5,3)
plt.title("K = 4",fontsize=16)
plt.xlabel("PL")
kmeans = KMeans(n_clusters=4)
features["labels"] = kmeans.fit_predict(features)
plt.scatter(features.PL[features.labels == 0],features.PW[features.labe
ls == 0])
plt.scatter(features.PL[features.labels == 1],features.PW[features.labe
ls == 1])
plt.scatter(features.PL[features.labels == 2],features.PW[features.labe
ls == 2])
plt.scatter(features.PL[features.labels == 3],features.PW[features.labe
ls == 3])

# Dropping the labels
features.drop(["labels"],axis=1,inplace=True)

plt.subplot(1,5,5)
plt.title("Original Labels",fontsize=16)
plt.xlabel("PL")
plt.scatter(iris.PL[iris.Species == "Iris-setosa"],iris.PW[iris.Species
 == "Iris-setosa"])
plt.scatter(iris.PL[iris.Species == "Iris-versicolor"],iris.PW[iris.Spe
cies == "Iris-versicolor"])
plt.scatter(iris.PL[iris.Species == "Iris-virginica"],iris.PW[iris.Spec
ies == "Iris-virginica"])

plt.subplots_adjust(top=0.8)
plt.show()

#Note: We took only two features : PetalLength(PL) and PetalWidth("PW")
```



## Conclusion

***It is clear from the visualization it is fit with 3 clusters. Except few data points, we can say prediction is identical to the original with labels. It means the Elbow was right.***

## Thanks!!

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: