Mail Traffic Analysis

- Student Name: Ryan Li Jian Tang

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import linregress
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

Part A

Task 1

In [2]:

```
df = pd.read_csv('CityPairs.csv')
```

Creating the dataframe that will contain the data for plotting the bar chart

Extracting the required Australian State Capital ports and the required data

In [3]:

```
wanted1 = df.groupby('AustralianPort').agg({'Mail_In_(tonnes)' : 'sum','Mail_Out_(tonnes)'
wanted1 = wanted1.rename(columns = {"Mail_In_(tonnes)" : "Total Mail In (tonnes)", "Mail_Ou
wanted1 = wanted1.reset_index()
wanted1 = wanted1[(wanted1['AustralianPort'] == 'Adelaide')|(wanted1['AustralianPort'] == '
wanted1
```

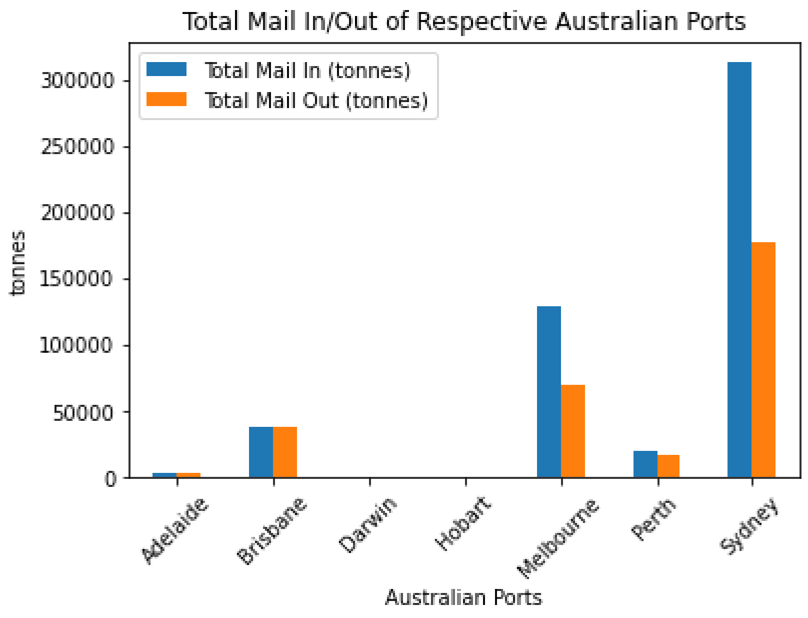
Out[3]:

	AustralianPort	Total Mail In (tonnes)	Total Mail Out (tonnes)
0	Adelaide	3151.792	3577.532
1	Brisbane	38971.895	37698.235
6	Darwin	401.673	589.191
9	Hobart	5.681	0.293
10	Melbourne	128609.327	70646.588
13	Perth	20788.444	17177.825
16	Sydney	312534.165	177177.967

Plot Bar Graph for both Total Mail In and Total Mail Out

In [4]:

```
bar = wanted1.plot.bar()
plt.title('Total Mail In/Out of Respective Australian Ports')
bar.set_xticklabels(wanted1['AustralianPort'], rotation = 45)
plt.xlabel('Australian Ports')
plt.ylabel('tonnes')
plt.legend()
plt.show()
```



Answers:

1.1. By looking at the bar chart above, it shows that out of all the selected ports, Sydney has currently received the largest amount of mails in compared to other cities.

1.2. No we cannot for some capitals, this is due to the fact that there is a huge discrepancy between the capitals with the highest amount of mails and the one with the least amount. Thus resulting in being unable to differentiate which of the capitals in the lower range have more mails. For example, between Darwin and Hobart it is impossible to tell which capital has the higher ground in terms of mails intake or outakes. Thus due to the scale of the axis it will not work for some capitals.

1.3. One obvious reason would be population. Comparing the population of Sydney and Hobart there is an overwhelming difference of around 5 million people (with Sydney at 5.23 million and Hobart at 0.2 million). Since there are a lot more people in Sydney that means there is a more probable chance that more mails will be exchanged on a daily basis in Sydney compared to Hobart. Furthermore as this is an acculumation of mails over the period of 31 years, it will result in in a staggering difference.

Task 2

Selecting required Capital Ports

In [5]:

```
Perth = df[df['AustralianPort'] == 'Perth']
Brisbane = df[df['AustralianPort'] == 'Brisbane']
```

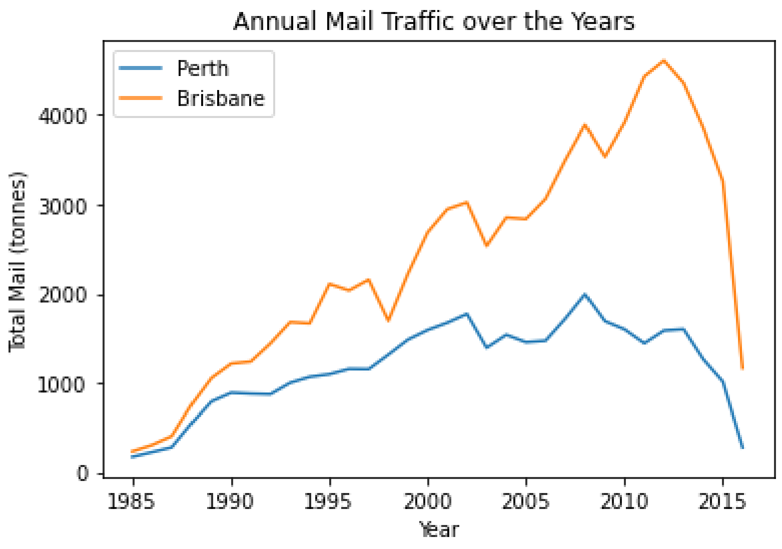
Creating the Dataframe that will be used for plotting the line graph

In [6]:

```
Perth = Perth.groupby('Year').agg({'Mail_Total_(tonnes)' : 'sum'})
Perth = Perth.reset_index()
Brisbane = Brisbane.groupby('Year').agg({'Mail_Total_(tonnes)' : 'sum'})
Brisbane = Brisbane.reset_index()
```

In [7]:

```
# Plotting line graph
plt.plot(Perth['Year'], Perth['Mail_Total_(tonnes)'], label = 'Perth')
plt.plot(Brisbane['Year'], Brisbane['Mail_Total_(tonnes)'], label = 'Brisbane')
plt.title('Annual Mail Traffic over the Years ')
plt.xlabel('Year')
plt.ylabel('Total Mail (tonnes)')
plt.legend()
plt.show()
```



In [8]:

```
Perth[Perth['Year'] <= 1990]
```

Out[8]:

	Year	Mail_Total_(tonnes)
0	1985	171.461
1	1986	224.049
2	1987	277.874
3	1988	545.534
4	1989	790.565
5	1990	887.316

In [9]:

```
Brisbane[Brisbane['Year'] <= 1990]
```

Out[9]:

	Year	Mail_Total_(tonnes)
0	1985	232.588
1	1986	303.250
2	1987	402.029
3	1988	756.031
4	1989	1052.270
5	1990	1213.883

In [10]:

```
Perth[Perth['Year'] == 2016]
```

Out[10]:

	Year	Mail_Total_(tonnes)
31	2016	277.265

In [11]:

```
Brisbane[Brisbane['Year'] == 2016]
```

Out[11]:

	Year	Mail_Total_(tonnes)
31	2016	1163.755

In [12]:

```
Perth.describe()
```

Out[12]:

	Year	Mail_Total_(tonnes)
count	32.000000	32.000000
mean	2000.500000	1186.445906
std	9.380832	490.796714
min	1985.000000	171.461000
25%	1992.750000	884.753000
50%	2000.500000	1288.560500
75%	2008.250000	1586.243000
max	2016.000000	1991.010000

In [13]:

```
Brisbane.describe()
```

Out[13]:

	Year	Mail_Total_(tonnes)
count	32.000000	32.000000
mean	2000.500000	2395.941563
std	9.380832	1257.965538
min	1985.000000	232.588000
25%	1992.750000	1389.400500
50%	2000.500000	2379.408000
75%	2008.250000	3319.829750
max	2016.000000	4611.991000

In [14]:

```
df['Month_num'].unique()
```

Out[14]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12], dtype=int64)
```

In [15]:

```
check = df[df['Year'] == 2016]
check['Month_num'].unique()
```

Out[15]:

```
array([1, 2, 3, 4, 5], dtype=int64)
```

Answers:

2.1. They both started at around 200 tonnes worth of total mail in 1985 and steadily increased until 1987. Afterwards proceeding to exponentially increase for the following years until 1990, shown by the sharp increase in the graph above and the huge difference from 1987 to 1990 for the 2 ports.

2.2. Looking at the Mail Total for 2016 and describe() for both ports, it shows that there is a problem in 2016. Where both Mail Total in 2016 were lower than at least 75% of the other years' Mail Total. This is a result of an incomplete data reading for both ports in 2016, looking at the cells above it shows that the data for 2016 was only recorded up until the 5th month of the year. This would result in significantly less Mail Total due to the lack of data for the rest of the months. Thus the result for this year would not paint an accurate potrait and thus can be considered as an outlier.

Part B1

Creating Dataframe for plotting:

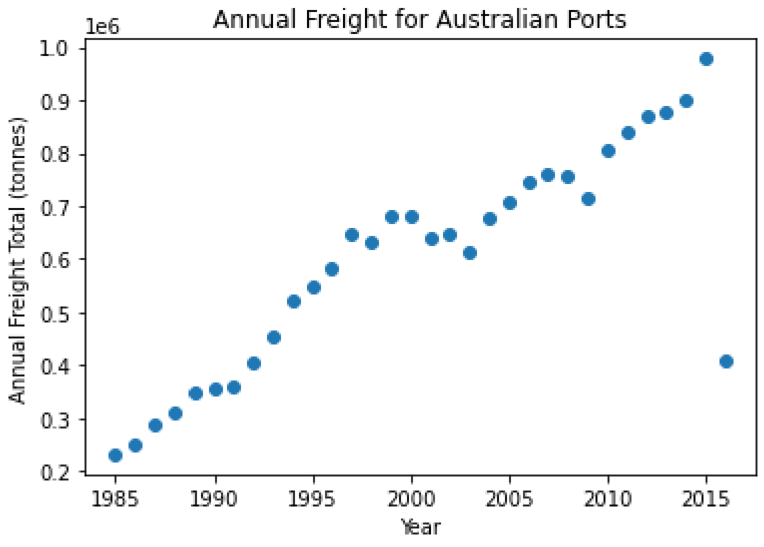
True_point is used for plotting the original unaltered points in the graphs.

In [42]:

```
PartB1 = df.groupby('Year').agg({'Freight_Total_(tonnes)':'sum'})
PartB1 = PartB1.reset_index()
True_point = PartB1[:]
```

In [43]:

```
plt.scatter(True_point['Year'],True_point['Freight_Total_(tonnes)'])
plt.xlabel('Year')
plt.ylabel('Annual Freight Total (tonnes)')
plt.title('Annual Freight for Australian Ports')
plt.show()
```



1. Yes it shows a clear linear pattern. It also further shows a fairly strong positive correlation.

In [18]:

```
PartB1.describe()
```

Out[18]:

	Year	Freight_Total_(tonnes)
count	32.000000	32.000000
mean	2000.500000	601114.898906
std	9.380832	207296.867236
min	1985.000000	231001.517000
25%	1992.750000	408189.088750
50%	2000.500000	643023.512500
75%	2008.250000	748914.308750
max	2016.000000	978847.712000

In [19]:

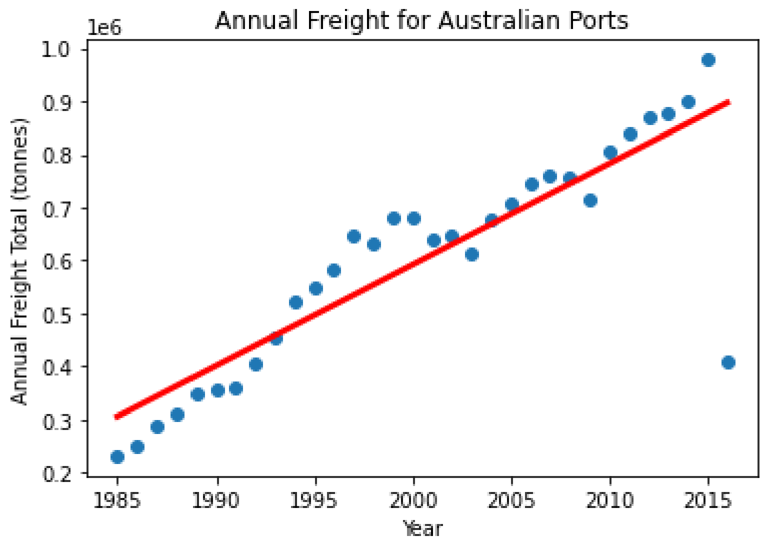
```
q1 = PartB1['Freight_Total_(tonnes)'].quantile(.25)
q3 = PartB1['Freight_Total_(tonnes)'].quantile(.75)
IQR = q3 - q1
# Using min and max to see if any of the data points can be considered an outlier
print("Are there any outliers? " + str((PartB1['Freight_Total_(tonnes)'].min() < q1 - 1.5*I
```

```
Are there any outliers? False
```

2. According to the IQR rule, there are no values that can be classified as an outlier in these results.

In [44]:

```
# Creating and plotting first regression model
gradient, intercept, r_value, p_value, std_err = linregress(PartB1['Year'], PartB1['Freight

line = [gradient*x + intercept for x in PartB1['Year']]
plt.plot(PartB1['Year'],line, color = 'red', linewidth = 3)
plt.scatter(True_point['Year'],True_point['Freight_Total_(tonnes)'])
plt.xlabel('Year')
plt.ylabel('Annual Freight Total (tonnes)')
plt.title('Annual Freight for Australian Ports')
plt.show()
```



3. The linear regression model seems to be a good fit, this can be seen by an equal spread of data being equidistant above and below the line. However, there is one data on the graph that is drastically further away from the line than the rest, which is the data for 2016. Showing at slightly less than half of the previous year in Annual Freight Total (tonnes). This data point can be considered as an outlier however according to the IQR rule it does not fall into the outlier zone.

In [45]:

```
print(gradient)
```

```
19096.915011913483
```

4. The gradient of the graph is an estimate of how fast the Annual Freight Total is increasing by each year. Furthermore it shows that on average, Annual Freight Total for Australian Ports will increase by roughly 19000 tonnes each year.

In [22]:

```
print("Annual Freight for 2020 is estimated to be " + str(gradient*2020 + intercept) + " to
```

```
Annual Freight for 2020 is estimated to be 973504.741638571 tonnes
```

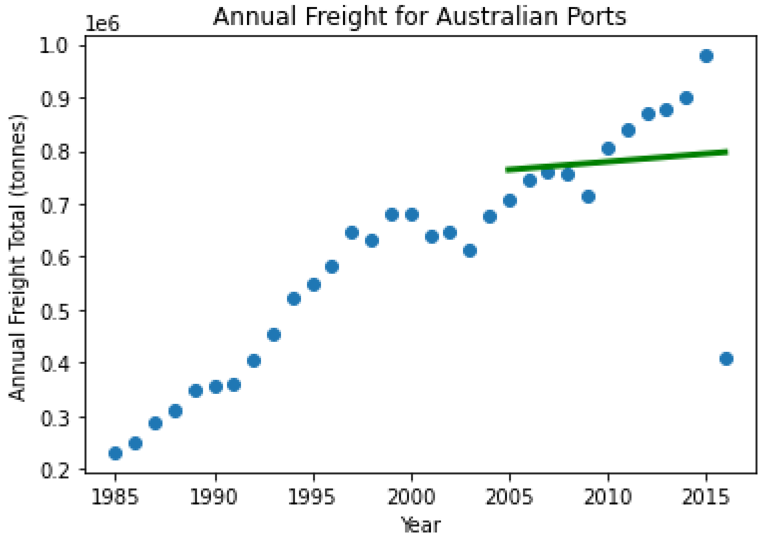
5. By using the formula of linear regression for this model, the linear regression model predicts that in 2020 the Annual Freight Total for Australian Ports will be roughly 970000 tonnes.

In [23]:

```
PartB1 = PartB1[PartB1['Year'] >= 2005]
```

In [24]:

```
# Creating plotting the second regression model
gradient, intercept, r_value, p_value, std_err = linregress(PartB1['Year'], PartB1['Freight

line = [gradient*x + intercept for x in PartB1['Year']]
plt.plot(PartB1['Year'],line, color = 'green', linewidth = 3)
plt.scatter(True_point['Year'],True_point['Freight_Total_(tonnes)'])
plt.xlabel('Year')
plt.ylabel('Annual Freight Total (tonnes)')
plt.title('Annual Freight for Australian Ports')
plt.show()
```



In [25]:

```
print("Annual Freight for 2020 is estimated to be " + str(gradient*2020 + intercept) + " to
```

```
Annual Freight for 2020 is estimated to be 809385.1952925408 tonnes
```

6. After doing the second linear regression model for the years after 2005, there is a large decrease in the prediction for 2020 of roughly 160000 tonnes in Annual Freight Total for the two different regression models. Comparing the two models, I would trust the initial regression model more. This is due to the fact that the initial regression model shows a much better line of best fit compared to the later model. The line of best fit for the second model is not a great fit as it does not maintain close to most of the points at equidistant levels. One of the reasons at to why that is, is due to the second model not containing enough data to be as reliable as the initial model. Therefore being more susceptible to anomalous or weird data, which happened in this case. The incomplete data for 2016 drastically affected the second regression model, painting an inaccurate picture.

Part B2

In [26]:

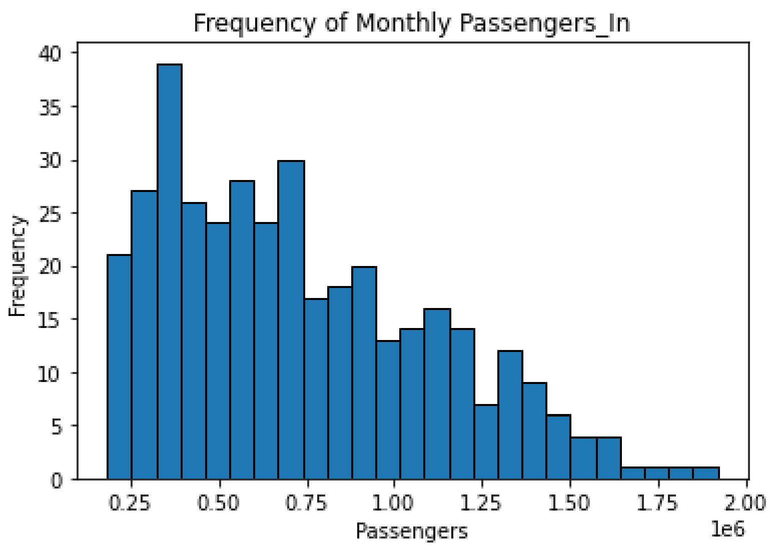
```
# Creating dataframe needed, with total number of Passenger In/Out
PartB2 = df.groupby('Month').agg({'Passengers_In':'sum', 'Passengers_Out' :'sum'})
PartB2 = PartB2.reset_index()
```

In [27]:

```
 def plot_histogram(variable : str) -> None:
    """ Plots a histogram based on the variable selected from the Dataframe PartB2
    """
    histogram = PartB2.hist(column= variable, bins=25, grid = False, ec = 'black')
    histogram = histogram[0]
    for x in histogram:
        x.set_xlabel('Passengers')
        x.set_ylabel('Frequency')
        x.set_title('Frequency of Monthly ' + variable)
```

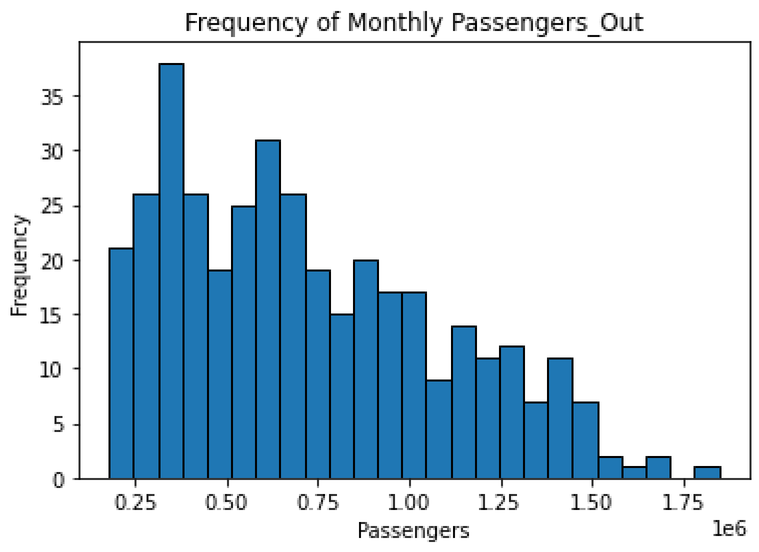
In [28]:

```
plot_histogram('Passengers_In')
```



In [29]:

```
plot_histogram('Passengers_Out')
```



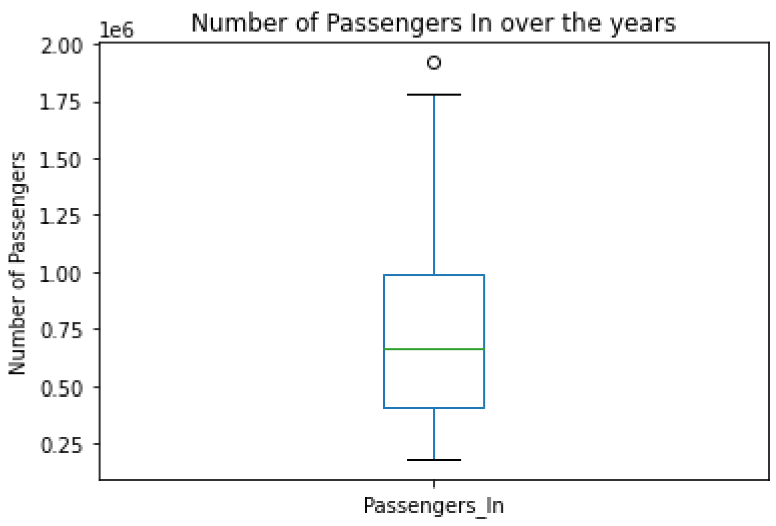
1. In both histograms shown above, they are both shown to be positively skewed. Shown by majority of the occurences happening on the left hand side. The most frequent bin being at slightly above 35 times with roughly (3.5 * 10^4) Passengers for both histograms. By just looking at the histograms above, it is hard to notice any outliers in the dataset except potentially in the second histogram. In the second histogram, the furthest bin on the right can be argued to be an outlier due to it being the only occurence at the highest amount accompanied with the fact that its neighbouring bin is empty as well.

In [30]:

```
box = PartB2.boxplot(column = 'Passengers_In',grid = False)
box.set_title('Number of Passengers In over the years')
box.set_ylabel('Number of Passengers')
```

Out[30]:

```
Text(0, 0.5, 'Number of Passengers')
```

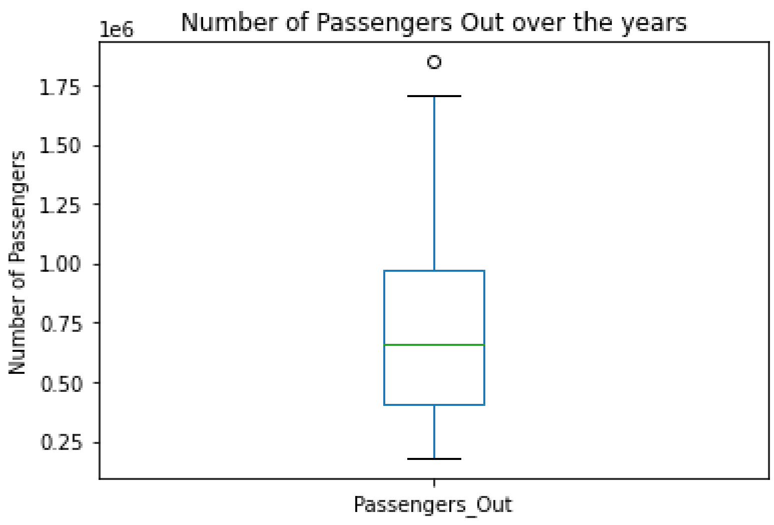


In [31]:

```
box = PartB2.boxplot(column = 'Passengers_Out',grid = False)
box.set_title('Number of Passengers Out over the years')
box.set_ylabel('Number of Passengers')
```

Out[31]:

```
Text(0, 0.5, 'Number of Passengers')
```



Using IQR rule to find the outliers.

In [32]:

```
PartB2.describe()
```

Out[32]:

	Month	Passengers_In	Passengers_Out
count	377.000000	3.770000e+02	3.770000e+02
mean	36769.169761	7.317297e+05	7.210568e+05
std	3316.906490	3.759759e+05	3.673378e+05
min	31048.000000	1.826730e+05	1.783190e+05
25%	33909.000000	4.079340e+05	4.037060e+05
50%	36770.000000	6.662590e+05	6.561840e+05
75%	39630.000000	9.944660e+05	9.759200e+05
max	42491.000000	1.918279e+06	1.847744e+06

In [33]:

```
Pas_In_Q1 = PartB2['Passengers_In'].quantile(.25)
Pas_In_Q3 = PartB2['Passengers_In'].quantile(.75)
Pas_In_IQR = Pas_In_Q3 - Pas_In_Q1
Pas_Out_Q1 = PartB2['Passengers_Out'].quantile(.25)
Pas_Out_Q3 = PartB2['Passengers_Out'].quantile(.75)
Pas_Out_IQR = Pas_Out_Q3 - Pas_Out_Q1
```

In [34]:

```
# Finding Outliers for Monthly Passengers In
PartB2[(PartB2['Passengers_In'] < Pas_In_Q1 - 1.5*Pas_In_IQR) | (PartB2['Passengers_In'] >
```

Out[34]:

	Month	Passengers_In	Passengers_Out
372	42370	1918279	1642059

In [35]:

```
# Finding Outliers for Monthly Passengers Out
PartB2[(PartB2['Passengers_Out'] < Pas_Out_Q1 - 1.5*Pas_Out_IQR) | (PartB2['Passengers_Out'
```

Out[35]:

	Month	Passengers_In	Passengers_Out
371	42339	1580055	1847744

2. According to the two boxplots above, in each plot there is one outlier which is shown by the dot. By using the IQR rule, the two outliers for the respective plots can be seen in the cells above.

3. In using a boxplot, we get a much easier visualisation of the data's visualisation compared to using the describe() method. Allowing us to understand the data much faster. In using a boxplot we are also able to determine immediately if there are any outliers present in the dataset, whereas in describe() we would have to manually incorporate the IQR rule to find any outliers. However, in using describe() we can find out the specifications of each specific outlier whereas it is not possible on the boxplot. Furthermore, a drawback in using a boxplot is that we are unable to determine the finer details of where each quartile lies at or what are the maximum and minimum values of the data set.

Part C

In [36]:

```
df1 = pd.read_csv('ClusteringData.csv')
df1
```

Out[36]:

	GDP per Capita	Suicide Rate
0	1.44178	0.44453
1	1.52733	0.41203
2	1.42666	0.14975
3	1.57744	0.35776
4	1.40598	0.41004
...	...	...
152	0.39499	0.06681
153	0.38227	0.07112
154	0.28123	0.11587
155	0.74719	0.17233
156	0.06831	0.09419

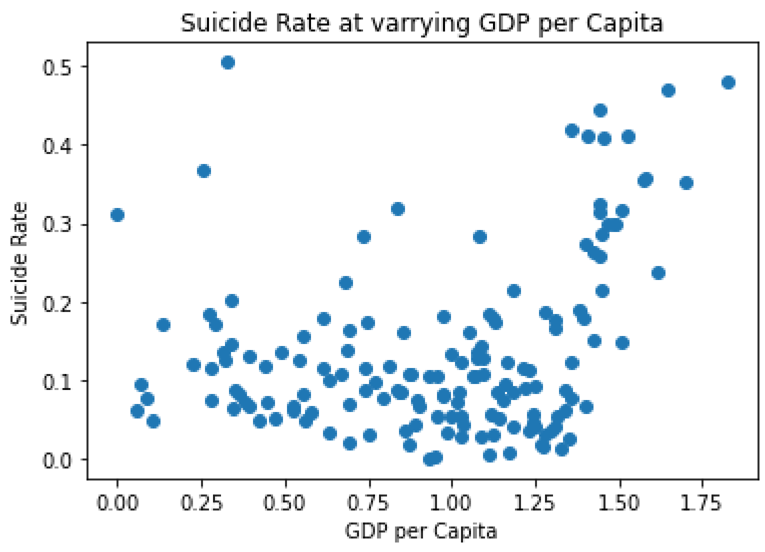
157 rows × 2 columns

In [37]:

```
# To see how the data are distributed
plt.scatter(df1['GDP per Capita'], df1['Suicide Rate'])
plt.xlabel('GDP per Capita')
plt.ylabel('Suicide Rate')
plt.title('Suicide Rate at varrying GDP per Capita')
```

Out[37]:

```
Text(0.5, 1.0, 'Suicide Rate at varrying GDP per Capita')
```



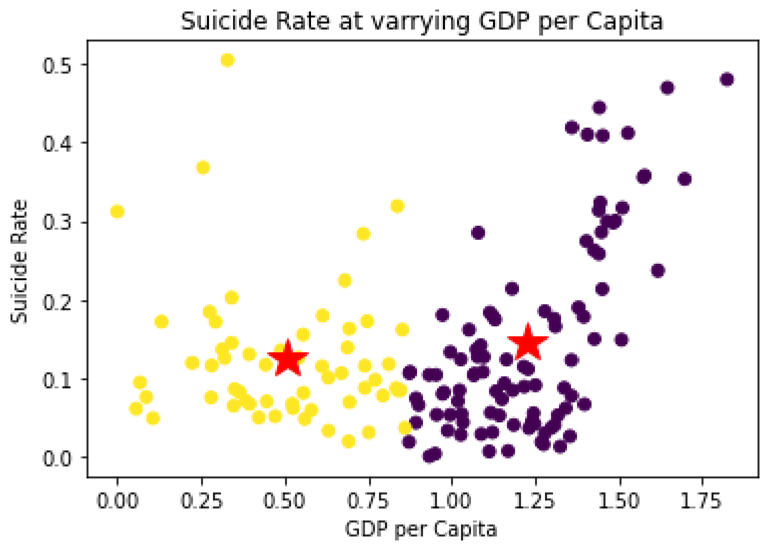
In [38]:

```
def plot_cluster(n : int, df, independent: str, dependent : str):
    """Plot cluster groups based on number of clusters desired on a set dataframe.
       Also returns coordinates of the cluster centers and the silhouette score of the cent
    """
    kmeans = KMeans(n_clusters=n).fit(df[[independent,dependent]])
    label = kmeans.predict(df)
    print(f'Silhouette Score(number of clusters = {n}): {silhouette_score(df,label)} \n')
    print('Coordinates of Cluster Centers are at: \n' + str(kmeans.cluster_centers_))

    plt.scatter(df[independent], df[dependent], c = kmeans.labels_)
    plt.plot(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],'k*', markersize = 2
    plt.xlabel(independent)
    plt.ylabel(dependent)
    # The next line can be removed, however it is placed for convenience
    plt.title('Suicide Rate at varrying GDP per Capita')
    plt.show()
```

In [39]:

```
plot_cluster(2,df1,'GDP per Capita', 'Suicide Rate')
```

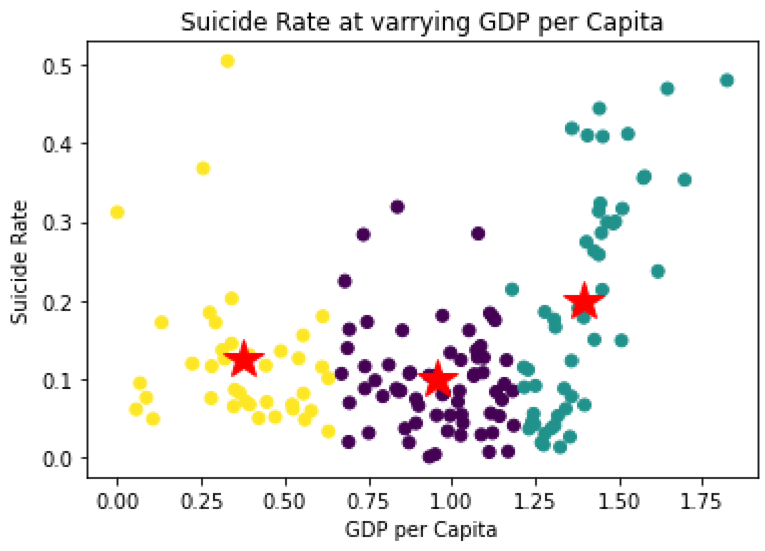
```
Silhouette Score(number of clusters = 2): 0.5627073439692872

Coordinates of Cluster Centers are at:
[[1.22383969 0.14566837]
 [0.50547186 0.12426153]]
```



In [40]:

```
plot_cluster(3,df1,'GDP per Capita', 'Suicide Rate')
```

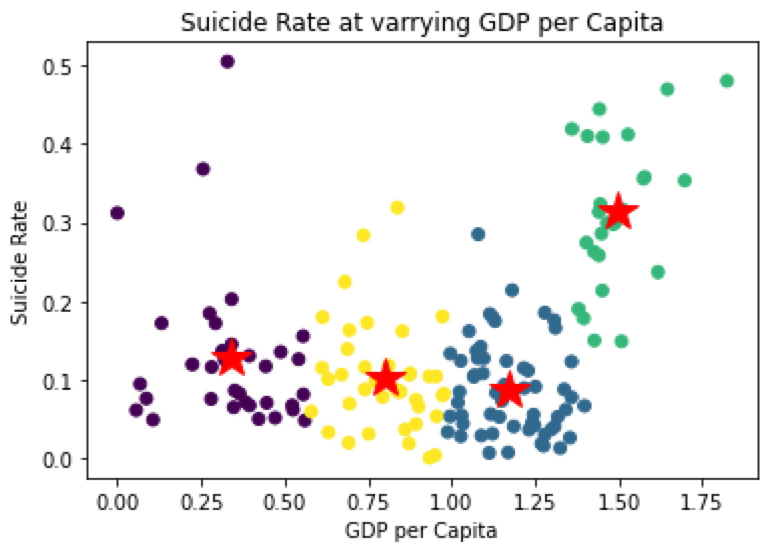
```
Silhouette Score(number of clusters = 3): 0.48907457426195083

Coordinates of Cluster Centers are at:
[[0.9581594  0.09887403]
 [1.39266098 0.19852078]
 [0.37273692 0.12455923]]
```



In [41]:

```
plot_cluster(4,df1,'GDP per Capita', 'Suicide Rate')
```

```
Silhouette Score(number of clusters = 4): 0.48507650147872766

Coordinates of Cluster Centers are at:
[[0.33734294 0.12857529]
 [1.17486897 0.08789017]
 [1.49354538 0.31375077]
 [0.8029459  0.10205692]]
```



Answers:

The best value of K based on my visualisations is using 3 centers. Comparing the first two graphs, when there are 3 centers the clusters are more well defined compared to when there are 2 centers. In regards to that, when there's only 2 centers it is harder to derive any useful information. Whereas, when there are 3 centers we can see a more obvious pattern where the clusters are seperated based on their GDP per capita levels. A lower, middle and upper tier earning area. Furthermore, it shows potentially that there is a higher occurence of suicide in areas with higher GDP per capita. On the other hand, when we increase the number of clusters by one more it becomes even more defined, however a bit too defined in my opinion. In my opinion, making it this defined will make possible implications for the future when utilising these centers for another set of data. Due to having too many clusters it might cause a group of data that exhibits the same characteristics to be split into multiple clusters due to the very fine distinctions in clusters, potentially giving too much information.

Challenge 1.

The Silhouette score is used to help decide whether the number of clusters used in a clustering technique is an optimal choice. This is based on a formula between neighbouring clusters, given as:

$$\frac{a-b}{max(a,b)}$$

a : *the average distance of all points within one cluster to its cluster center.*

b *: the average distance of all points of the cluster used in* ***a*** *to the next nearest cluster.*

The ranges of values that are attainable are in a range from -1 to 1. A value close to 1 means that that there is a distinct difference between the locations of cluster centers and the data are allocated to the right cluster. A value close to 0 would mean that there is no distinct difference between clusters, a result of the centers almost stacking on each other due to having too many centers. A value close to -1 would mean that the data are assigned to the wrong clusters but have a very distinct difference between the clusters.

Generally, the higher the Silhouette score that is attained, the better the fit the number of clusters chosen are.

1.1. A score of 0.02 means that the centers are extremely close to each other, almost stacking. However, being positive implies that the samples are allocated to the right clusters.

1.2. A score of -0.06 would imply that the cluster centers are close to each other as well, almost indistinguishable. However, a negative value would imply that there are samples that are applied to the wrong cluster and there is a more appropriate cluster available.

1.3. A score of 0.97 implies a really good score, possibly the best optimal situation. This shows that there is a huge distinction between the clusters and that they are far apart from each other. Furthermore it shows that all samples have been allocated to the correct cluster.

1.4. A score of -0.9 implies strongly that a huge majority of the samples might have been allocated to the wrong cluster and needs to be fixed. However, a large value means that there is a huge distinction between the clusters and that they are seperated.

2.

According to the 3 graphs above and their silhouette score, it would seem that the best number of clusters is 2. At a silhouette value of 0.5627, which is much higher than when k is 3 or 4. Therefore, showing a better fit of samples to their clusters.

Challenge 2

1. Square Transformation
2. Yes, it is a good idea. One reason being is that you are able to see a clear pattern right now compared to before. Thus, showing that there is now a negative correlation in the dataset. Furthermore, with the data being more organised a meaningful linear regression model can now be fitted onto the data whereas previously the linear regression model would only display a straight line, where y = 0. This is caused by the distributions of the data; where for each point on the x-axis there is always a pair of data that is equidistant from y = 0, thus cancelling each other in the regression model.