SAYANTAN RAHA

Roll # : BAI09056

IIMB - BAI09 - Assignment 1

Out[1]: Toggle on/off Code

In [2]:
```
import warnings
warnings.filterwarnings('ignore')

%load_ext rpy2.ipython
```

In [3]:
```
import pandas as pd
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Q1

Q1 - a

In [4]:
```
cgpaList = [3.36, 1.56, 1.48, 1.43, 2.64, 1.48, 2.77, 2.20, 1.38, 2.84,
            1.88, 1.83, 1.87, 1.95, 3.43, 1.28, 3.67, 2.23, 1.71, 1.68,
            2.57, 3.74, 1.98, 1.66, 1.66, 2.96, 1.77, 1.62, 2.74, 3.35,
            1.80, 2.86, 3.28, 1.14, 1.98, 2.96, 3.75, 1.89, 2.16, 2.07]

print("List of cgpa scores: {}".format(cgpaList))
```

```
List of cgpa scores: [3.36, 1.56, 1.48, 1.43, 2.64, 1.48, 2.77, 2.2, 1.38, 2.84, 1.88, 1.83, 1.87, 1.95, 3.43, 1.28, 3.67,
2.23, 1.71, 1.68, 2.57, 3.74, 1.98, 1.66, 1.66, 2.96, 1.77, 1.62, 2.74, 3.35, 1.8, 2.86, 3.28, 1.14, 1.98, 2.96, 3.75, 1.8
9, 2.16, 2.07]
```

MEAN

The Mean of a list of numbers are defined as the sum of all the numbers in the list divided by the number of elements of the list.

n = number of elements in list

$$Mean = \frac{(\sum_{k=1}^{n} x_k)}{n}$$

In [5]:
```
meanList = sum(cgpaList) / len(cgpaList)
print("Mean of CGPA: {}".format(meanList))
```

```
Mean of CGPA: 2.2652499999999995
```

MEDIAN

The Median is the mid Value of the list of numbers. Here n (number of items in list) is even hence median is average of

$$\frac{(n)}{2}$$

and

$$\frac{(n+2)}{2}$$

element of the list.

First we will sort all items in the list in ascending order, then take average of 20 and 21 element of the list to find the median of the list

In [6]:
```
cgpaList.sort()
print("Sorted list is : {}".format(cgpaList))
print("")
print("20th Element of list is: {}".format(cgpaList[19]))
print("21th Element of list is: {}".format(cgpaList[20]))
print("")

print("Median of list is: {}".format((cgpaList[20] + cgpaList[20])/2))
```

```
Sorted list is : [1.14, 1.28, 1.38, 1.43, 1.48, 1.48, 1.56, 1.62, 1.66, 1.66, 1.68, 1.71, 1.77, 1.8, 1.83, 1.87, 1.88, 1.8
9, 1.95, 1.98, 1.98, 2.07, 2.16, 2.2, 2.23, 2.57, 2.64, 2.74, 2.77, 2.84, 2.86, 2.96, 2.96, 3.28, 3.35, 3.36, 3.43, 3.67,
3.74, 3.75]

20th Element of list is: 1.98
21th Element of list is: 1.98

Median of list is: 1.98
```

MODE

Mode is the most frequent occurring element in the list. Sort the elements and count the most frequesnt ones. The mode(s) of the list is:

In [7]:
```
print("Mode is:");
print(pd.Series(cgpaList).mode())
```

```
Mode is:
0    1.48
1    1.66
2    1.98
3    2.96
dtype: float64
```

Standard Deviation

The standard deviation of a list of numbers is calculated by:

$$\sigma = \sqrt{\frac{\sum_{k=1}^{n}(x_k - \bar{x})^2}{n-1}}$$

Instead of using n, we use n - 1. This is called Bessel's correction, which is required due to **downward bias**. When we compute the average from sample, we underestimate the numerator.

Another possible explanation is, since we compute the mean from the sample, we loose **one degree of freedom**. Hence the denominator is n-1

In [8]:
```
print("Sample Mean: {}".format(np.mean(cgpaList)))
print("Sum of Squared Difference from Mean: {}".format((sum((cgpaList  - np.mean(cgpaList)) ** 2))))
print("Standard Deviation: {}".format(((sum((cgpaList  - np.mean(cgpaList)) ** 2))/(len(cgpaList) - 1))**0.5))
```

```
Sample Mean: 2.26525
Sum of Squared Difference from Mean: 22.1391975
Standard Deviation: 0.7534399317591488
```

Q1 - b

Percentile

Position corresponding to percentile (x) is given by the following formula:

$$P_x = \frac{x * (n+1)}{100}$$

Value at position is computed as:

$$Val_x = Val_{P_x} + Fractionalpartof P_x * (Value_{P_{x+1}} - Value_{P_x})$$

In [9]:
```
import math
cgpaList.sort()
#print(cgpaList)
pos = 90* (len(cgpaList) + 1)/100
dec = round(math.modf(pos)[0], 2)
val = cgpaList[int(pos) - 1] + dec * (cgpaList[int(pos)] - cgpaList[int(pos) - 1])
print("Position corresponding to Percentile 90: {}".format(math.floor(pos)))
print("Value corresponding to Percentile 90: {}".format(val))

pos = 95* (len(cgpaList) + 1)/100
dec = round(math.modf(pos)[0], 2)
val = cgpaList[int(pos) - 1] + dec * (cgpaList[int(pos)] - cgpaList[int(pos) - 1])
print("Position corresponding to Percentile 95: {}".format(math.floor(pos)))
print("Value corresponding to Percentile 95: {}".format(round(val,3)))
```

```
Position corresponding to Percentile 90: 36
Value corresponding to Percentile 90: 3.423
Position corresponding to Percentile 95: 38
Value corresponding to Percentile 95: 3.737
```

Q1 - c

IQR

IQR is the range between **Percentile 25** and **Percentile 75**. The formula for computing percentile is same as above.

In [10]:
```
cgpaList.sort()
pos = 25 * (len(cgpaList) + 1)/100
dec = round(math.modf(pos)[0], 2)
val25 = cgpaList[int(pos) - 1] + dec * (cgpaList[int(pos)] - cgpaList[int(pos) - 1])
print("Position corresponding to Percentile 25: {}".format(math.floor(pos)))
print("Value corresponding to Percentile 25: {}".format(round(val25,3)))

pos = 75* (len(cgpaList) + 1)/100
dec = round(math.modf(pos)[0], 2)
val75 = cgpaList[int(pos) - 1] + dec * (cgpaList[int(pos)] - cgpaList[int(pos) - 1])
print("Position corresponding to Percentile 75: {}".format(math.floor(pos)))
print("Value corresponding to Percentile 75: {}".format(round(val75,3)))

print("IQR: {}".format(val75 - val25))
```

```
Position corresponding to Percentile 25: 10
Value corresponding to Percentile 25: 1.665
Position corresponding to Percentile 75: 30
Value corresponding to Percentile 75: 2.855
IQR: 1.19
```

Q1 - d

Plotting a histogram or computing Skew will both provide whether the distribution has right tail or not

SKEW Skewness is computed by the following equation (**Pearson moment coefficient of skewness**):

$$g_1 = \frac{\sum_{k=1}^{n} \frac{(x_k - \bar{x})^3}{n}}{\sigma^3}$$

$$\sigma = StandardDeviation$$

For samples with n onservation the formula is adjusted as follows:

$$G_1 = \sqrt{\frac{n * (n-1)}{n-2}} * g_1$$

If G1 < 0 it is left Skewed If G1 > 0 it is right Skewed For G1 near 0, the distribution is considered to be symmetric

In [11]:
```
x_bar = np.mean(cgpaList)
numerator = sum((cgpaList - x_bar) ** 3) / len(cgpaList)
sigma = np.std(cgpaList)

skewness = numerator / sigma ** 3
print("Mean: {}".format(x_bar))
print("Numerator: {}".format(numerator))
print("Sigma: {}".format(sigma))
print("Skewness: {}".format(round(skewness, 3)))

print("This distribution is slightly Right Tailed")
```

```
Mean: 2.26525
Numerator: 0.23074712128124997
Sigma: 0.7439623226346883
Skewness: 0.56
This distribution is slightly Right Tailed
```

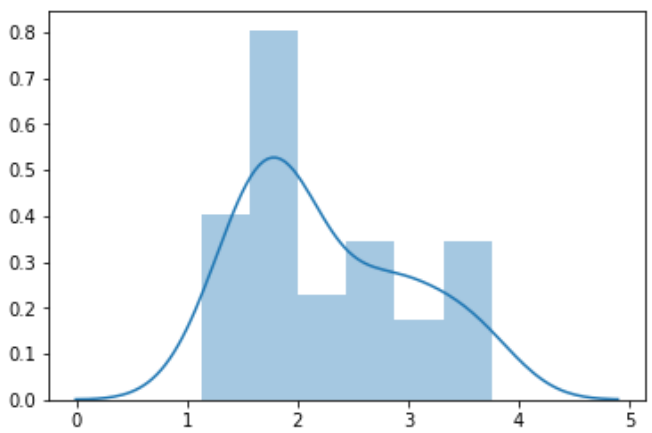
Q1 - e

The Optimal number of bins is computed using the following formula: **N = 1 =3.322 * Log10 (n)** where n = number of onservations

In [12]:
```
N = 1 + 3.322* np.log10(len(cgpaList))
print("Number of bins: {}".format(int(N)))
sns.distplot(cgpaList, bins= int(N))
```

```
Number of bins: 6
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f987ab262e8>
```



Q2

Q2 - a

In [13]:
```
data = pd.read_excel("./Chapter 2 BKB.xlsx", sheet_name='DKB Bank Data')
#data.head()
text = """
"The data has Nominal Variable (e.g. Loan_Type, Gender, Accomodation Type etc.) ,
Continuous variables (e.g. Monthly Salary, Balance in Savings Account)
and Discrete Continuous variables (e.g. No of Years in Job etc.)"
"""
print(text)
```

```
"The data has Nominal Variable (e.g. Loan_Type, Gender, Accomodation Type etc.) ,
Continuous variables (e.g. Monthly Salary, Balance in Savings Account)
and Discrete Continuous variables (e.g. No of Years in Job etc.)"
```

In [14]:
```
#data.describe(include='all').transpose()
```

In [15]:
```
plt.figure(figsize = (10, 5))

plt.subplot(2, 2, 1)

sns.countplot( y = 'Loan Type',
          data = data)

plt.subplot(2, 2, 2)

sns.countplot( y = 'Gender',
          data = data)

plt.subplot(2, 2, 3)

sns.countplot( y = 'Marital Status',
          data = data)

plt.subplot(2, 2, 4)

sns.countplot( y = 'Accomodation Type',
          data = data)


plt.tight_layout()
```

Observations from above graphs

- Most Loans are Home Loans
- Most Loan Applicants are Male
- Proportion of Married people apply for loans are higher
- Most people applying for loan live in rented accomodation

Observations above have to be validated by hypothesis test to prove statistical significance

Accomodation Type vs Loan Type

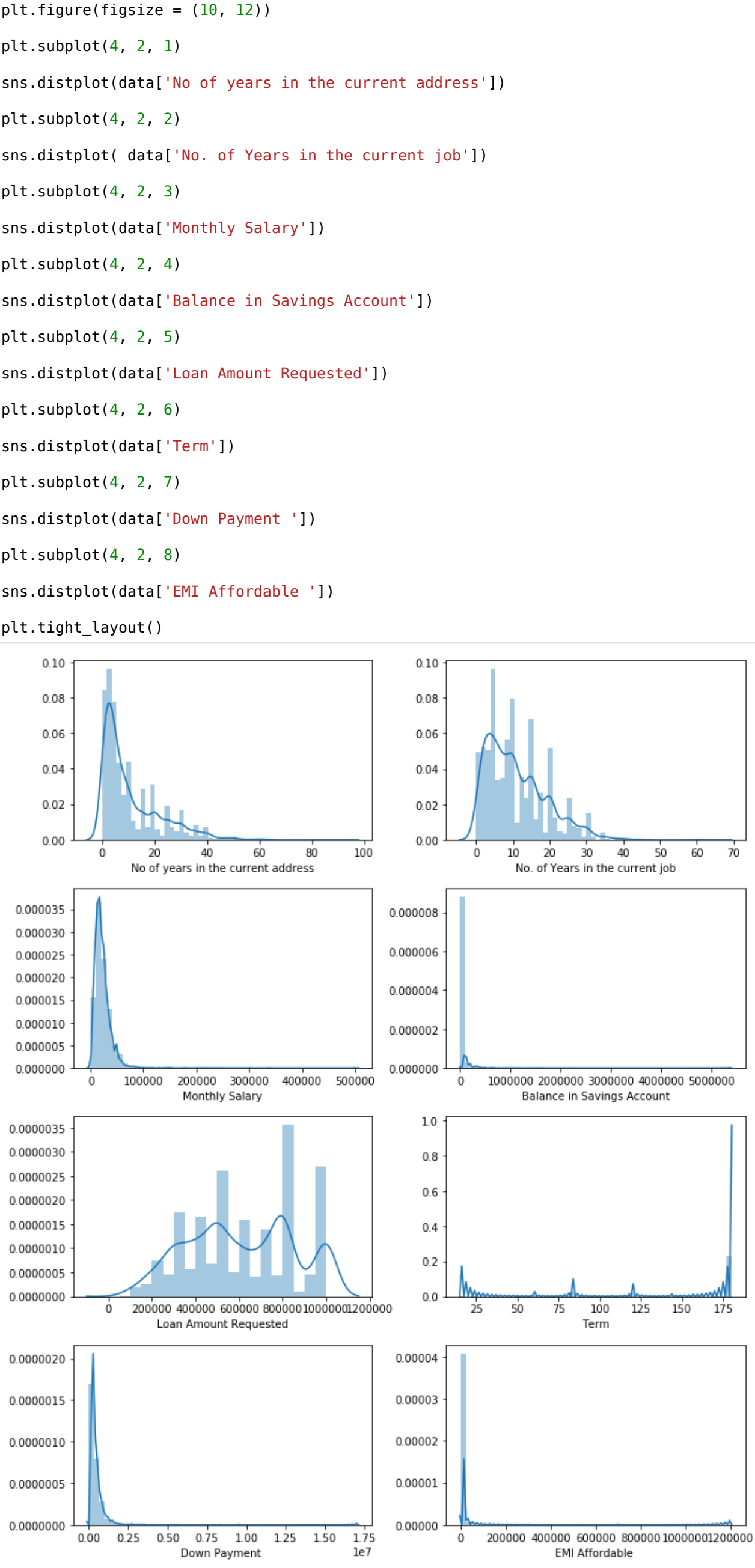
The Data below shows there is a correlation between Loan Type and Accomodation type. This has to be further validated with statistical hypothesis test.

In [16]:
```
print(data[['Loan Type', 'Marital Status' ,'Gender', 'Accomodation Type']].apply(lambda x :
                                                                    pd.factorize(x)[0]).corr(method =
                                                                                             'spearman'))
data[['Accomodation Type','Loan Type']].groupby(['Accomodation Type','Loan Type']).size()
```

```
                  Loan Type  Marital Status    Gender  Accomodation Type
Loan Type          1.000000       -0.016972  0.003260          -0.130769
Marital Status    -0.016972        1.000000  0.066943          -0.003602
Gender             0.003260        0.066943  1.000000           0.016295
Accomodation Type -0.130769       -0.003602  0.016295           1.000000
```

Out[16]:
```
Accomodation Type  Loan Type
Company Provided   Home Improvement       2
                   Home Loan             86
Family Other       Home Improvement      62
                   Home Loan            914
Owned              Home Improvement     258
                   Home Loan            771
Rented             Home Improvement      30
                   Home Loan           1741
dtype: int64
```

In [17]:
```
plt.figure(figsize = (10, 12))

plt.subplot(4, 2, 1)

sns.distplot(data['No of years in the current address'])

plt.subplot(4, 2, 2)

sns.distplot( data['No. of Years in the current job'])

plt.subplot(4, 2, 3)

sns.distplot(data['Monthly Salary'])

plt.subplot(4, 2, 4)

sns.distplot(data['Balance in Savings Account'])

plt.subplot(4, 2, 5)

sns.distplot(data['Loan Amount Requested'])

plt.subplot(4, 2, 6)

sns.distplot(data['Term'])

plt.subplot(4, 2, 7)

sns.distplot(data['Down Payment '])

plt.subplot(4, 2, 8)

sns.distplot(data['EMI Affordable '])

plt.tight_layout()
```

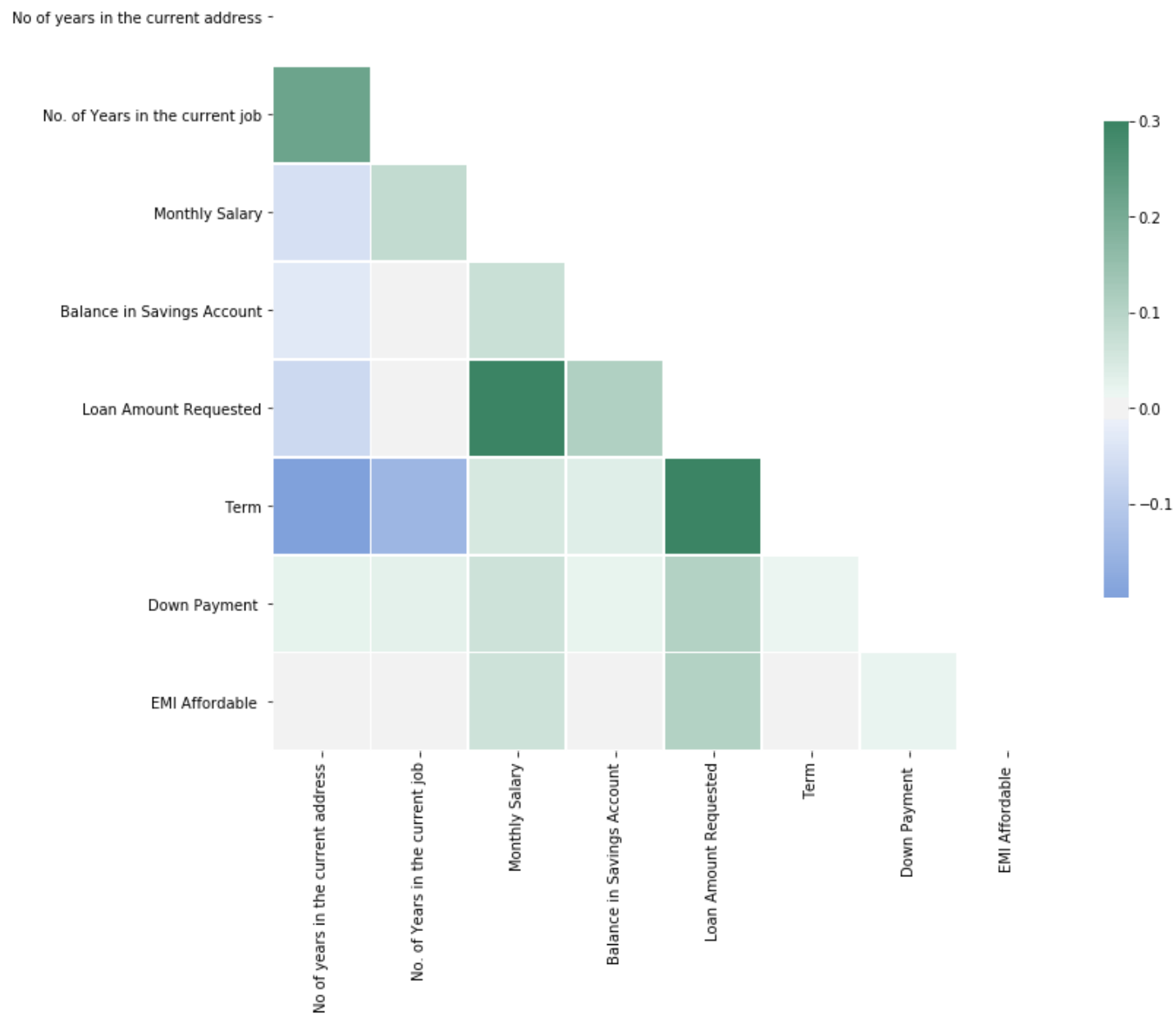


Observations

- Number of years in Current Address - Right Skewed
- No. of Years in current Job - Right Skewed. Jumps at 5, 10, 15, 20, 25, 30, implying people have a habit of rounding off to nearest 5 year mark
- Salary is right skewed, most of salaries are withing 100K, with a few high value outliers
- Balance in saving account - Mostly less than 50K, with a few high values
- Loan Amount - Multimodal distribution. The values are rounded off more towards 200K, 300K, 400K, 500K, 600K, 700K, 800K 1000K
- Loan tenure - Most apply for 180 Months of loan tenure
- Fewer proportion of people can afford high EMI

In [18]:
```
corr = data[['No of years in the current address', 'No. of Years in the current job',
      'Monthly Salary', 'Balance in Savings Account', 'Loan Amount Requested', 'Term',
     'Down Payment ', 'EMI Affordable ']].corr(method='pearson')

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(12, 12))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(255, 150, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9871ec6be0>
```



In [19]:
```
data[['Accomodation Type','No. of Years in the current job', 'No of years in the current address',
      'Down Payment ', 'Term', 'Balance in Savings Account', 'Monthly Salary', 'Loan Amount Requested']].groupby(
    ['Accomodation Type']).agg('mean')
```

Out[19]:

Accomodation Type	No. of Years in the current job	No of years in the current address	Down Payment	Term	Balance in Savings Account	Monthly Salary	Loan Amount Requested
Company Provided	13.590909	11.659091	430372.818182	160.670455	52078.568182	16838.806818	662881.818182
Family Other	11.215164	16.846311	453250.704918	164.195697	31620.655738	23363.690574	646135.420082
Owned	12.651118	12.990282	430270.357629	140.511176	19325.789116	20295.480078	499773.082604
Rented	9.632976	5.717109	411493.360248	169.322981	37665.495200	23845.810277	649441.705251

Observations

- Applicants who live on rented apartments do tend to spend less time on same job and on same address, where as the other applicant types spend more time on same address and job. The corrleation of time spent on current address and job shows positive correlation
- Applicants owning homes / properties seem to apply on average smaller amount of loan and for lesser tenure
- Applicants living on Company Provided properties seems to have higher bank balance with lesser monthly salary on average

All the above observations need to be validated statistically via hypothesis tests.

Q2 - b

Formula and rules for Mean, Median, Mode, Standard Deviation and Skew is provided in answers for Qs1

The Kurtosis is given by the equation:

$$Kurtosis = \frac{\sum_{k=1}^{n} \frac{(x_k - \bar{x})^4}{n}}{\sigma^4}$$

$$\sigma = StandardDeviation$$

Monthly Salary

In [20]:
```
print("Mean: {}".format(np.sum(data['Monthly Salary']) / data['Monthly Salary'].shape[0]))
print("Total elements: {}".format(data['Monthly Salary'].shape[0]))
```

```
Mean: 22618.98447204969
Total elements: 3864
```

In [21]:
```
data.sort_values('Monthly Salary', inplace = True)
data.reset_index(inplace = True, drop = True)

x = data['Monthly Salary']
print("Median: {}".format((x[x.shape[0] / 2 - 1] +
                           x[x.shape[0] / 2])/2))
```

```
Median: 19000.0
```

In [22]:
```
print("Mode :\n {}".format(data['Monthly Salary'].mode()))
```

```
Mode :
 0    15000
dtype: int64
```

In [23]:
```
x_bar = np.mean(data['Monthly Salary'])

numerator = sum((data['Monthly Salary'] - x_bar) ** 3) / data['Monthly Salary'].shape[0]
sigma = np.std(data['Monthly Salary'])

skewness = numerator / sigma ** 3
print("Calculation for computing Skew: ")

print("Numerator: {}".format(numerator))
print("Sigma: {}".format(sigma))
print("Skewness: {}".format(round(skewness, 3)))
```

```
Calculation for computing Skew:
Numerator: 61538297730342.375
Sigma: 19780.762269971405
Skewness: 7.951
```

In [24]:
```
x_bar = np.mean(data['Monthly Salary'])
numerator = sum((data['Monthly Salary'] - x_bar) ** 4) / data['Monthly Salary'].shape[0]
sigma = np.std(data['Monthly Salary'])

skewness = numerator / sigma ** 4
print("Numerator: {}".format(numerator))
print("Sigma: {}".format(sigma))
print("Kurtosis: {}".format(round(skewness, 3)))
```

```
Numerator: 2.0575588117373972e+19
Sigma: 19780.762269971405
Kurtosis: 134.394
```

Balance in Savings Account

In [25]:
```
print("Mean: {}".format(np.sum(data['Balance in Savings Account']) / data['Balance in Savings Account'].shape[0]))
print("Total elements: {}".format(data['Balance in Savings Account'].shape[0]))
```

```
Mean: 31582.945910973085
Total elements: 3864
```

In [26]:
```
data.sort_values('Balance in Savings Account', inplace = True)
data.reset_index(inplace = True, drop = True)

x = data['Balance in Savings Account'].sort_values()
print("Median: {}".format((x[x.shape[0] / 2 - 1] +
                           x[x.shape[0] / 2 ])/2))
```

```
Median: 6357.5
```

In [27]:
```
print("Mode :\n {}".format(data['Balance in Savings Account'].mode()))
```

```
Mode :
 0    0
dtype: int64
```

In [28]:
```
x_bar = np.mean(data['Balance in Savings Account'])

numerator = sum((data['Balance in Savings Account'] - x_bar) ** 3) / data['Balance in Savings Account'].shape[0]
sigma = np.std(data['Balance in Savings Account'])

skewness = numerator / sigma ** 3
print("Calculation for computing Skew: ")

print("Numerator: {}".format(numerator))
print("Sigma: {}".format(sigma))
print("Skewness: {}".format(round(skewness, 3)))
```

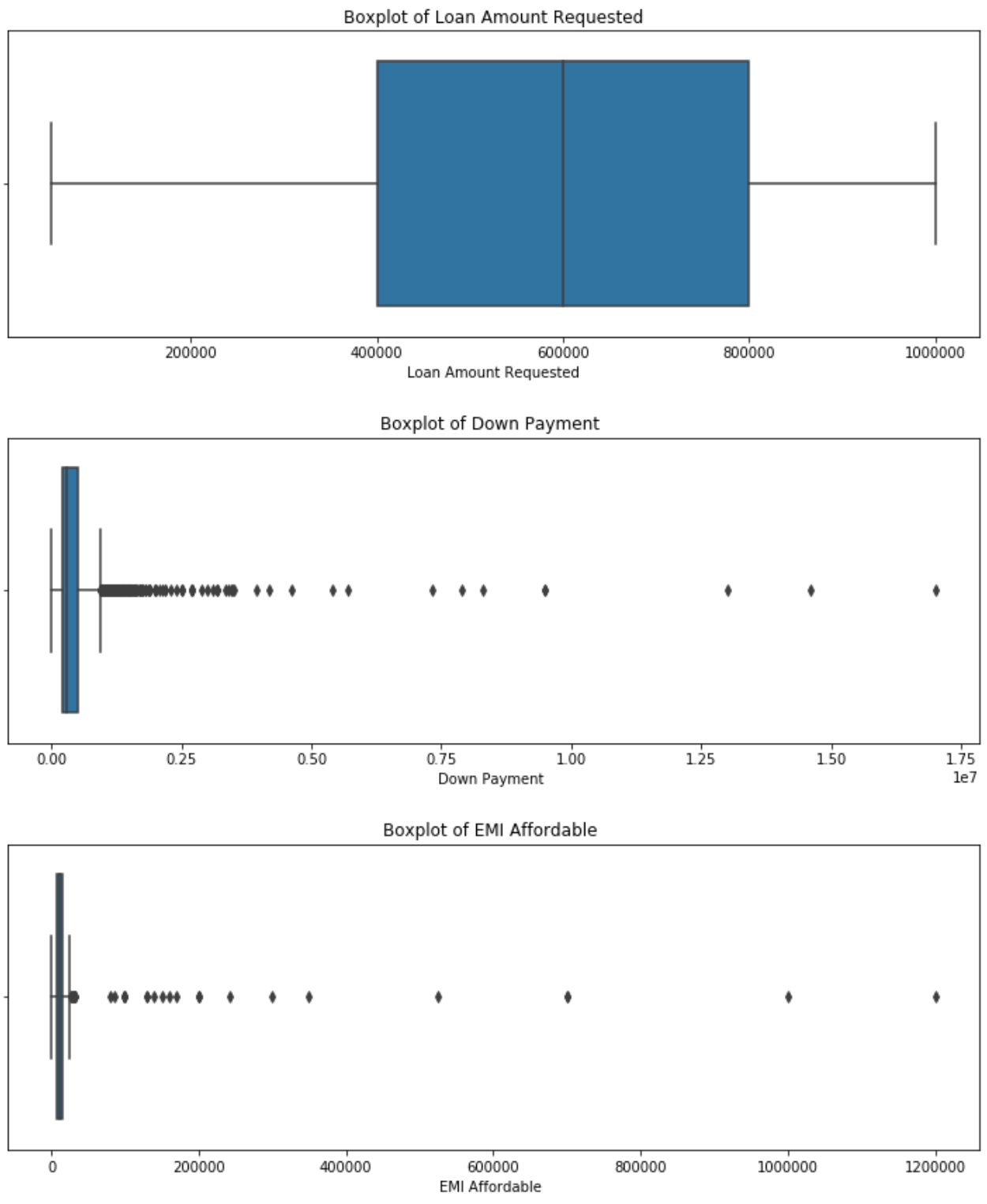
```
Calculation for computing Skew:
Numerator: 4.8671220825693384e+16
Sigma: 127482.44692023737
Skewness: 23.492
```

In [29]:
```
x_bar = np.mean(data['Balance in Savings Account'])
numerator = sum((data['Balance in Savings Account'] - x_bar) ** 4) / data['Balance in Savings Account'].shape[0]
sigma = np.std(data['Balance in Savings Account'])

skewness = numerator / sigma ** 4
print("Numerator: {}".format(numerator))
print("Sigma: {}".format(sigma))
print("Kurtosis: {}".format(round(skewness, 3)))
```

```
Numerator: 2.2645418005570515e+23
Sigma: 127482.44692023737
Kurtosis: 857.391
```

Q2 - c

In [30]:
```
plt.figure(figsize = (10, 12))

for i, source in enumerate(['Loan Amount Requested', 'Down Payment ', 'EMI Affordable ']):
    plt.subplot(3, 1, i + 1)
    sns.boxplot(data[source])

    # Label the plots
    plt.title('Boxplot of %s' % source)

plt.tight_layout(h_pad = 2.5)
```



Observations:

- Loan Amount does not have outliers
- Downpayment and EMI Affordable has outliers

Q2 - d

Continuous variables are the following:

- Monthly Salary
- Loan Amount Requested
- Down Payment
- EMI Affordable

In [31]:
```
colLst = ['Monthly Salary', 'Loan Amount Requested', 'Down Payment ', 'EMI Affordable ']

for col in colLst:
    print("Column {} has skew : {}".format(col, sp.stats.skew(data[col])))
    print("Column {} has Kurtosis : {}".format(col, sp.stats.kurtosis(data[col])))
```

```
Column Monthly Salary has skew : 7.950902138083359
Column Monthly Salary has Kurtosis : 131.39408769367247
Column Loan Amount Requested has skew : 0.0073934349150113765
Column Loan Amount Requested has Kurtosis : -1.018197531983543
Column Down Payment  has skew : 13.041420458301284
Column Down Payment  has Kurtosis : 257.61353462521816
Column EMI Affordable  has skew : 25.533691859591546
Column EMI Affordable  has Kurtosis : 760.4961457368507
```

Hence in descending order of Skewness and Kurtosis the columns can be arranged in following order:

```
- EMI Affordable - has highest skew
- Down Payment - Number 2 in skewness
- Monthly Salary = Number 3 in skewness
- Loan Amount Requested has no skewness
```

Q3

Q3 - a

Z stat is computed by following equation:

$$z = (x - \mu)/\sigma$$

In [32]:
```
mu = 68
sigma = 14
x = 90

z = (x - mu) / sigma
p90 = sp.stats.norm.cdf(z, 0, 1)

print("Z Statistic for mean: {}, variance: {}, x: {}, Z-Stat: {}".format(mu, sigma, x, z))
print("Corresponding CDF: {}".format(p90))

print("Proportion of orders delivered after 90 mins = 1 - CDF(90): {} %".format(round((1-p90)* 100, 2)))
```

```
Z Statistic for mean: 68, variance: 14, x: 90, Z-Stat: 1.5714285714285714
Corresponding CDF: 0.9419584331306725
Proportion of orders delivered after 90 mins = 1 - CDF(90): 5.8 %
```

Q3 - b

For 99% order fulfillment within 90 mins, Target Mean is given by equation:

$$\mu = 90 - Z_c * \sigma$$

In [33]:
```
z = sp.stats.norm.ppf(.99, 0, 1)
print("Critical Zc: {}".format(z))
mu = 90 - z * sigma
print("Mean Delivery Time :{}".format(mu))
```

```
Critical Zc: 2.3263478740408408
Mean Delivery Time :57.43112976342823
```

Q3 - c

To meet 99% orders and to maintain same mean and variance the committed time should be calculated as following:

$$x = \mu + Z_c * \sigma$$

In [34]:
```
mu = 68
x = z * sigma + mu
print("For Mean {}, variance {}, committed time is {}".format(mu, sigma, x))
```

```
For Mean 68, variance 14, committed time is 100.56887023657177
```

Q4

Q4 - a

For exponential distribution:

$$Mean = \frac{1}{\lambda}$$

$$\lambda = Rate of decay$$

Probability of Missing Flight :: P(X >= 25) = 1 - P(X <= 25)

In [35]:
```
lmbda = 1.0/20
time_left = 40 - 15
print("lambda : {}".format(lmbda))
print("Time Left for Security check for Ms. TKW is {} mins".format(time_left))
p25 = sp.stats.expon.cdf(time_left, scale = 1 / lmbda)
print("Probability of missing flight: {}".format(round((1 - p25), 3)))
```

```
lambda : 0.05
Time Left for Security check for Ms. TKW is 25 mins
Probability of missing flight: 0.287
```

Q4 - b

Probability of Waiting for more than 40 min :: P(X >= 40) = 1 - P(X <= 40)

In [36]:
```
time_left = 40
p40 = sp.stats.expon.cdf(40, scale = 1 / lmbda)
print("Probability of Ms. TKW waiting for {} mins at security check: {}".format(40, round((1 - p40), 3)))
```

```
Probability of Ms. TKW waiting for 40 mins at security check: 0.135
```

Q4 - c

Exponential Distributions are memory less.

$$P(X \geq 20 + 20 | X \geq 20) = P(X \geq 20) = \exp^{-\lambda * 20}$$

$$\lambda = Rate of decay$$

In [37]:
```
PXgt20 = np.exp(-1*lmbda * 20)
print("Probability that she would wait for another 20 mins : {}".format(round(PXgt20, 3)))
```

```
Probability that she would wait for another 20 mins : 0.368
```

Q4 - d

Equation to find t is given as follows:

$$t = -1 * \frac{1}{\lambda} * \ln(0.01)$$

In [38]:
```
t = -1 * (1/lmbda) * np.log(0.01)
print("To ensure she catches flight 99% of times she should reach airport {} hours / {} mins before departure".format(round(
```

```
To ensure she catches flight 99% of times she should reach airport 1.54 hours / 92.1 mins before departure
```

Q5

Q5 - 1

We will use T-test for this hypothesis test as the population standard deviation is unknown.

T-statistic equation:

$$T = \frac{\bar{x} - \mu}{S/\sqrt{n}}$$

where S = Sample Sample SD

In [39]:
```
data = pd.read_excel("./Jayalaxmi Agro Case Data .xlsx", sheet_name="App Usage Data")

d6 = data[data.Year >= '2015-10-01 00:00:00']
print("We assume that the correct data is in column D6")
#sns.distplot(d6.D6, bins= 5)

mu = 30
xbar = d6.D6.mean()
S = d6.D6.std()
n = d6.shape[0]

print("Mean of accessing D6 info:{}".format(xbar))
print("Sigma of accessing D6 info:{}".format(S))
print("Number of observations of accessing D6 info:{}".format(n))


print("""Since the population standard deviation is not known, we will compute same from data.\n
We will use T-Test for the hypothesis""")

ho = "ho = Mean of accessing D6 <= 30"
ha = "ha = Mean of accessing D6 > 30"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a Right sided T-Test")

tstat = (xbar - mu) / (S / n ** 0.5)

print("Value of T-statistics: {}".format(tstat))

alpha = 0.05
print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {} @ df {}".format(alpha, -1* sp.stats.t.ppf(alpha, df = n-1), n-1))
print("p-value:{} is less than alpha({})".format(1 - sp.stats.t.cdf(tstat, df = n-1), alpha))
print("Hence we can reject the NULL Hypothesis (ho)")
```

```
We assume that the correct data is in column D6
Mean of accessing D6 info:40.863046476139324
Sigma of accessing D6 info:19.21537805197856
Number of observations of accessing D6 info:20
Since the population standard deviation is not known, we will compute same from data.

We will use T-Test for the hypothesis
Null hypothesis: ho = Mean of accessing D6 <= 30
Alternate hypothesis: ha = Mean of accessing D6 > 30
This is a Right sided T-Test
Value of T-statistics: 2.5282365298960063
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : 1.7291328115213678 @ df 19
p-value:0.010240954188212248 is less than alpha(0.05)
Hence we can reject the NULL Hypothesis (ho)
```

The claim disease 6 (leaf curl) was accessed at least 30 times every month on average since October 2015 is statistically significant.

Q5 - 2

We will use Z-test for this hypothesis test for proportions.

Z-statistic equation:

$$Z = \frac{\hat{p} - p}{\sqrt{\frac{p*(1-p)}{n}}}$$

where

$$\hat{p} = EstimatedSampleproportion$$

p = Expected Proportion

The above is valid for large n (sample size) and if

$$n * p * (1 - p) >= 10$$

In [40]:
```
d6users = data.D6.sum()
totusers = data.D1.sum() + data.D2.sum() + data.D3.sum() + data.D4.sum() + data.D5.sum() + data.D6.sum() + data.D7.sum() + d

print("d6 users {}".format(d6users))
print("total Disease application users (Sum of D1..D11): {}".format(totusers))
p = d6users / totusers
print("proportion of d6 users {}".format(p))
n = data.shape[0]
print("n = {}".format(n))
print("n * p * (1- p)= {}".format(n * p * (1-p)))
#print("n * q =() {}".format(n * (1 - p)))

print("n * p * (1 - p) is less than 10, hence the power of the Z test for proportions may not be correct. ")


print("\nProceeding to calculate the metrics even though the assumptions are incorrect.")

phat = .15

S = (p * (1 - p) / n) ** 0.5
print("Standard Error: {}".format(S))

zstat = (phat - p) / S
print("Z-Stat {}".format(zstat))

ho = "Proportion of D6 users <= 15%"
ha = "Proportion of D6 users > 15%"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a Right sided T-Test")
```

```
print("alpha/ Significance: {}".format(alpha))
print("Significant Z-value at alpha - {} is : {}".format(alpha, -1* sp.stats.norm.ppf(alpha)))
print("p-value:{} is greater than alpha({})".format(1 - sp.stats.norm.cdf(zstat), alpha))
print("Hence we can retain the NULL Hypothesis (ho) but as the assumptions are not valid, the test results may not be valid"
```

```
d6 users 856.806242628309
total Disease application users (Sum of D1..D11): 5713.156874519798
proportion of d6 users 0.14997071871938142
n = 24
n * p * (1- p)= 3.0595080539081665
n * p * (1 - p) is less than 10, hence the power of the Z test for proportions may not be correct.

Proceeding to calculate the metrics even though the assumptions are incorrect.
Standard Error: 0.07288103955710223
Z-Stat 0.00040176815254719557
Null hypothesis: Proportion of D6 users <= 15%
Alternate hypothesis: Proportion of D6 users > 15%
This is a Right sided T-Test
alpha/ Significance: 0.05
Significant Z-value at alpha - 0.05 is : 1.6448536269514729
p-value:0.49983971770134217 is greater than alpha(0.05)
Hence we can retain the NULL Hypothesis (ho) but as the assumptions are not valid, the test results may not be valid
```

Q5 - 3

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

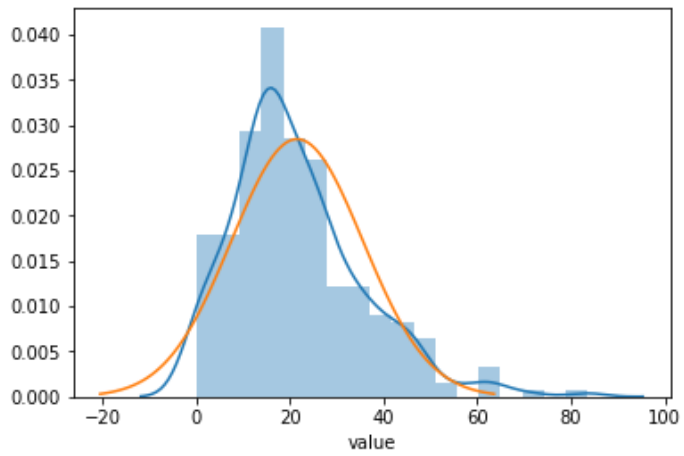
S1, S2 are the Sample SD, n1, n2 are the sample lengths

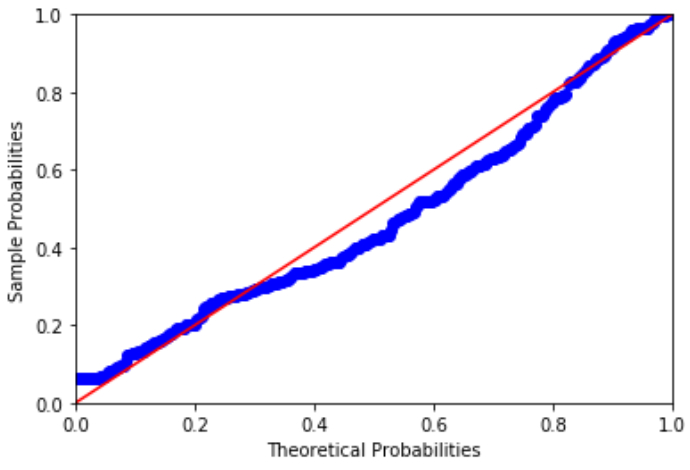
In [41]:
```
jun15jun16 = data.loc[(data.Year >= "2015-06-01 00:00:00") & (data.Year <= "2016-06-01 00:00:00"),
                      "Number of users"]
n1 = jun15jun16.shape[0]
x1 = jun15jun16.mean()
S1 = jun15jun16.std()

print("Total Usage data for Time >= June 2015 & Time < Jul-2016 :: len (n2) = {}, mean (x1) = {}, se (S1) = {}".format(n1, x

jun16may17 = data.loc[(data.Year > "2016-06-01 00:00:00"), "Number of users"]#Number of users
n2 = jun16may17.shape[0]
x2 = jun16may17.mean()
S2 = jun16may17.std()
print("Total Usage data for Time >= Jul-2016 :: len (n2) = {}, mean (x2) = {}, se (S2) = {}".format(n2, x2, S2))

#print(x1)
#print(x2)
#print(S1)
#print(S2)

print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SE {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("DF {}".format(df))

tstat = ((x2 - x1) - 0) /(Su)
print("T-stat {}".format(tstat))

ho = "Null Hypothesis: ho = x2 - x1 <= 0"
ha = "Alternate hypothesis ha = x2 - x1 > 0"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a right sided T-Test")

print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , -1* sp.stats.t.ppf(alpha, df = df)))
print("p-value:{} is less than alpha({} for t-stat {})".format(1 - sp.stats.t.cdf(tstat, df = df), alpha, tstat))
print("Hence we can reject the NULL Hypothesis (ho)")
print("We have proven statistically that number of users have increased over the years")
```

```
Total Usage data for Time >= June 2015 & Time < Jul-2016 :: len (n2) = 13, mean (x1) = 198.53846153846155, se (S1) = 73.049
77228417094
Total Usage data for Time >= Jul-2016 :: len (n2) = 11, mean (x2) = 409.3636363636364, se (S2) = 219.79775828123124
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 69.29932393687176
DF 11
T-stat 3.0422399938161893
Null hypothesis: Null Hypothesis: ho = x2 - x1 <= 0
Alternate hypothesis: Alternate hypothesis ha = x2 - x1 > 0
This is a right sided T-Test
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : 1.7958848187036696
p-value:0.005600843093913066 is less than alpha(0.05 for t-stat 3.0422399938161893)
Hence we can reject the NULL Hypothesis (ho)
We have proven statistically that number of users have increased over the years
```

Q5 - 4

In [42]:
```
tblmean = data[["D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11"]].mean()
tblstd = data[["D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11"]].std()
usgdata = data[["D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11"]]
#print(tblmean)
#print(tblstd)

#print(usgdata.melt())


sns.distplot(usgdata.melt().value)
import matplotlib.mlab as mlab
import math
import statsmodels.api as sm

mu = np.mean(usgdata.melt().value)
sigma = np.std(usgdata.melt().value)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
plt.plot(x,mlab.normpdf(x, mu, sigma))
pp_x = sm.ProbPlot(usgdata.melt().value, fit=True)
pp_x.ppplot(line='45')

plt.show()
```





In [43]:
```
print("Its close to normal distribution, with close SD.")
print("Performing Anova")
ho = "ho = means of groups are same"
ha = "ha = means of groups are not same"
print("NULL Hypothesis: {}".format(ho))
print("Alternate Hypothesis: {}".format(ha))

# Anova by hand

data2 = usgdata.melt()
k = len(np.unique(data2.variable))
N = data2.shape[0]
n = data2.groupby('variable').size()[0]
#print(k)
DFbetween = k - 1
DFwithin = N - k
DFtotal = N - 1
SSbetween = (sum(data2.groupby('variable').sum()['value']**2)/n) \
    - (data2['value'].sum()**2)/N

sum_y_squared = sum([value**2 for value in data2['value'].values])
SSwithin = sum_y_squared - sum(data2.groupby('variable').sum()['value']**2)/n

SStotal = sum_y_squared - (data2['value'].sum()**2)/N
MSbetween = SSbetween/DFbetween

MSwithin = SSwithin/DFwithin
F = MSbetween/MSwithin
print("F-Statistic: {}".format(F))
#print(sp.stats.f.cdf(F, DFbetween, DFwithin))
print("Critical F {}, for DFBetween {} and DFWithin {}".format(sp.stats.f.ppf(.95, DFbetween, DFwithin), DFbetween, DFwithin
#print(sp.stats.f_oneway(usgdata.D1, usgdata.D2, usgdata.D3, usgdata.D4, usgdata.D5, usgdata.D6,
#                  usgdata.D7, usgdata.D8, usgdata.D9, usgdata.D10, usgdata.D11))

print("P-Value: 1.54e-05 is less that 0.05")
print("We reject the NUll Hypothesis")
```

```
Its close to normal distribution, with close SD.
Performing Anova
NULL Hypothesis: ho = means of groups are same
Alternate Hypothesis: ha = means of groups are not same
F-Statistic: 4.287407508171504
Critical F 1.8682470831185125, for DFBetween 10 and DFWithin 253
P-Value: 1.54e-05 is less that 0.05
We reject the NUll Hypothesis
```

There is a statistically significant difference in the average number of times various diseases are accessed

Q5 - 5

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

S1, S2 are the Sample SD, n1, n2 are the sample lengths

In [44]:
```
jun15jun16 = data.loc[(data.Year >= "2015-06-01 00:00:00") & (data.Year <= "2016-06-01 00:00:00"),
                      "Usage"]
n1 = jun15jun16.shape[0]
x1 = jun15jun16.mean()
S1 = jun15jun16.std()
print("Usage data for Time >= June 2015 & Time < Jul-2016 :: len (n2) = {}, mean (x1) = {}, se (S1) = {}".format(n1, x1, S1))

jun16may17 = data.loc[(data.Year > "2016-06-01 00:00:00"), "Usage"]#Number of users
n2 = jun16may17.shape[0]
x2 = jun16may17.mean()
S2 = jun16may17.std()

print("Usage data for Time >= Jul-2016 :: len (n2) = {}, mean (x2) = {}, se (S2) = {}".format(n2, x2, S2))

print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
ho = "The usage means x2 - x1 <= 0"
ha = "x2 - x1 > 0"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a right sided T-Test")


Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SE Adjusted: {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("DF: {}".format(df))

tstat = ((x2 - x1) - 0) /(Su)
print("T-stat {}".format(tstat))


print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , -1* sp.stats.t.ppf(alpha, df = df)))
print("p-value:{} is less than alpha({})".format(1 - sp.stats.t.cdf(tstat, df = df), alpha))
print("Hence we can reject the NULL Hypothesis (ho)")
print("\n")
print("In Q5 - 3, we have already proven statistically that number of total users have increased over the years")
print("Above we have already proven statistically that number of app users have increased over the years")

print("\nCorrelation Analysis:")
print("Since both are Quantitative variables we will use Pearson Coefficient.")
#print(data[["Usage", "Number of users"]].corr())

rho = np.sum((data.Usage.values - data.Usage.mean())*
             (data["Number of users"].values - data["Number of users"].mean())) /(
    (data.shape[0] - 1) * data["Number of users"].std() * data.Usage.std())
print("Correlation : {}".format(rho))
print("Correlation is positive and non-zero")

print('\n\nWe will perform hypothesis test for Significance of correlation.')
ho ="ho : rho == 0"
ha = "ha : rho != 0"
print("Null Hypothesis: {}".format(ho))
print("Alternate Hypothesis: {}".format(ha))
print("This is a two sided T-test")

tstat = rho *((22)/(1 - rho **2))**0.5
print("T-stat {}".format(tstat))
tcrit = -1* sp.stats.t.ppf(0.025, df = 22)
print("T-critical {} at alpha 0.05".format(tcrit))
print("T-stat < T-critical for Two sided T-test, hence we retain Null hypothesis (ho), the correlation between App users and
```

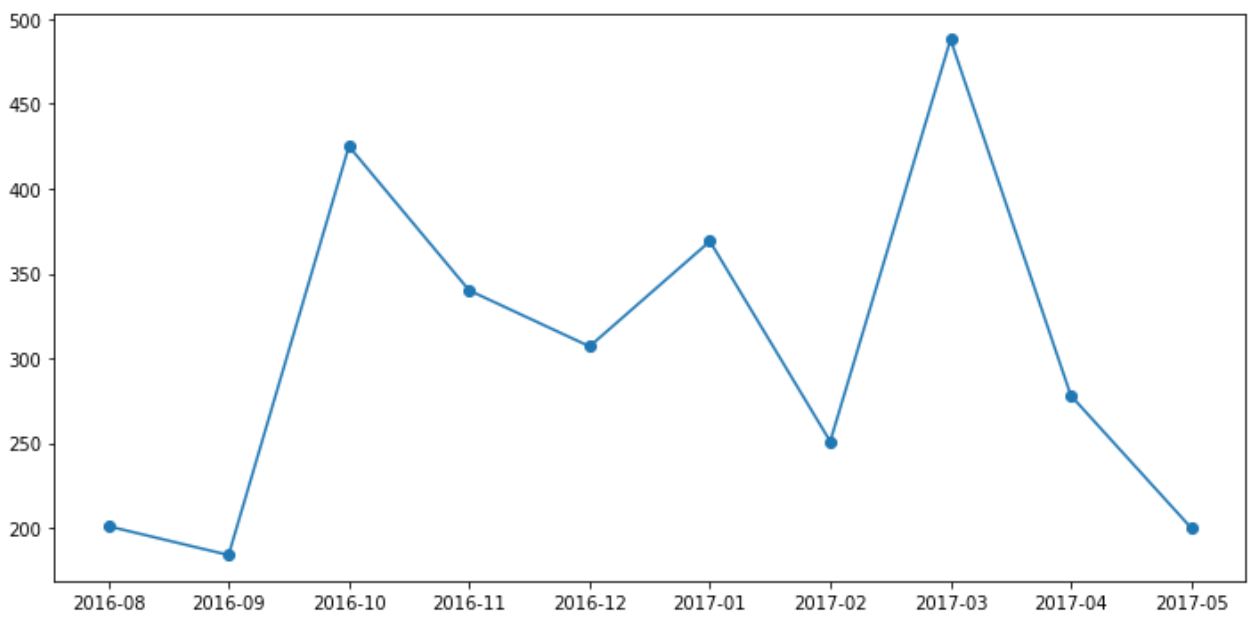
```
Usage data for Time >= June 2015 & Time < Jul-2016 :: len (n2) = 13, mean (x1) = 152.15384615384616, se (S1) = 98.276689465
5477
Usage data for Time >= Jul-2016 :: len (n2) = 11, mean (x2) = 296.6363636363636, se (S2) = 99.8982209323797
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
Null hypothesis: The usage means x2 - x1 <= 0
Alternate hypothesis: x2 - x1 > 0
This is a right sided T-Test
SE Adjusted: 40.62250691274703
DF: 21
T-stat 3.556711007345043
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : 1.7207429028118777
p-value:0.0009325540627762585 is less than alpha(0.05)
Hence we can reject the NULL Hypothesis (ho)


In Q5 - 3, we have already proven statistically that number of total users have increased over the years
Above we have already proven statistically that number of app users have increased over the years

Correlation Analysis:
Since both are Quantitative variables we will use Pearson Coefficient.
Correlation : 0.3503700784608535
Correlation is positive and non-zero


We will perform hypothesis test for Significance of correlation.
Null Hypothesis: ho : rho == 0
Alternate Hypothesis: ha : rho != 0
This is a two sided T-test
T-stat 1.7546032832085956
T-critical 2.073873067904015 at alpha 0.05
T-stat < T-critical for Two sided T-test, hence we retain Null hypothesis (ho), the correlation between App users and total
users is not statistically significant.
```

It has been proven that Total users and total appusers have increased over the two periods, but correlation between the two is not statisticaaly relevant.

Q5 - 6

In [45]:
```
newAppRel = data.loc[(data.Year >= "2016-08-01 00:00:00"), ["Year","Usage"]]
newAppRel["Year"] = newAppRel.Year.dt.strftime("%Y-%m")
plt.figure(figsize=(10,5))
plt.plot("Year", "Usage", data=newAppRel, marker='o')
plt.tight_layout()
print("From the graph it is clear that October - 2016 the app usage shifts")
```

```
From the graph it is clear that October - 2016 the app usage shifts
```



Q5 - 7

We will perform T-test between means for users accessing for Disease information during favourable and unfavourable periods for the 5 Diseases provided (with weather data) for the 2 Regions.

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

S1, S2 are the Sample SD, n1, n2 are the sample lengths

In [46]:
```
# Define Functions
def prepareData(dharwad, source, ignoreT = False,
                ignoreH = False, T2 = True, H2 = False,
                lowerT = 20, higherT = 24, lowerH = 80, higherH = None):
    print("Disease type: {}".format(source))
    dharwad['d1_weather'] = 0
    if ((ignoreH == False) & (ignoreT == False) & (T2 == True) & (H2 == False)):
        dharwad.loc[(dharwad["Temperature"] >= lowerT) & (dharwad["Temperature"] <= higherT)
                    & (dharwad["Relative Humidity"] > lowerH),
                    'd1_weather'] = 1
    elif ((ignoreH == False) & (ignoreT == False) & (T2 == False) & (H2 == False)):
        dharwad.loc[(dharwad["Temperature"] > lowerT)
                    & (dharwad["Relative Humidity"] > lowerH),
                    'd1_weather'] = 1
    elif ((ignoreH == True) & (ignoreT == False) & (T2 == True)):
        dharwad.loc[(dharwad["Temperature"] >= lowerT)
                    & (dharwad["Temperature"] <= higherT),
                    'd1_weather'] = 1
    elif ((ignoreH == False) & (ignoreT == False) & (T2 == True) & (H2 == True)):
        dharwad.loc[(dharwad["Temperature"] >= lowerT)
                    & (dharwad["Temperature"] <= higherT)
                    & (dharwad["Relative Humidity"] >= lowerH)
                    & (dharwad["Relative Humidity"] <= higherH),
                    'd1_weather'] = 1


    x1 = dharwad.loc[dharwad['d1_weather'] == 1, source].mean()
    S1 = dharwad.loc[dharwad['d1_weather'] == 1, source].std()
    n1 = dharwad.loc[dharwad['d1_weather'] == 1, source].shape[0]

    #print(x1, S1, n1)

    x2 = dharwad.loc[dharwad['d1_weather'] == 0, source].mean()
    S2 = dharwad.loc[dharwad['d1_weather'] == 0, source].std()
    n2 = dharwad.loc[dharwad['d1_weather'] == 0, source].shape[0]

    #print(x2, S2, n2)
    return(x1, S1, n1, x2, S2, n2)


def pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2):
    print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
    Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
    #print(Su)

    df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
    #print("Degrees of freedom: {}".format(df))
    print("X1 (Mean of users checking Disease in fav condition) {}, SE {}, Len {}".format(x1, S1, n1))
    print("X2 (Mean of users checking Disease in un-fav condition) {}, SE {}, Len {}".format(x2, S2, n2))

    tstat = ((x1 - x2) - 0) /(Su)
    #print(tstat)

    ho = "The means of diseases access: x1 - x2 <= 0"
    ha = "x1 - x2 > 0"

    print("Null hypothesis: {}".format(ho))
    print("Alternate hypothesis: {}".format(ha))
    print("This is a right sided T-Test")

    #print("alpha/ Significance: {}".format(alpha))
    print("Significant t-value at alpha - {} and df: {} is : {}".format(alpha, df , -1* sp.stats.t.ppf(alpha, df = df)))
    tcrit = -1* sp.stats.t.ppf(alpha, df = df)


    if(tstat <= tcrit):
        print("T Statistics:{} is less than T Critical: {} at alpha({})".format(tstat, tcrit, alpha))
        print("Hence we can retain the NULL Hypothesis (ho)")
        print("")
    else:
        print("T Statistics:{} is gt than T Critical: {} at alpha({})".format(tstat, tcrit, alpha))
        print("Hence we can reject the NULL Hypothesis (ho)")
        print("")
```

In [47]:
```
dharwad = pd.read_excel("./Jayalaxmi Agro Case Data .xlsx", sheet_name="Dharwad_weather")

print("Analysing for Dharwad")
source = 'D1'
ignoreT = False
ignoreH = False
T2 = True
H2 = False
lowerT = 20
higherT = 24
lowerH = 80
higherH = None
#dharwad.head()
alpha = 0.1

x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)

source = 'D2'
ignoreT = False
ignoreH = False
T2 = True
H2 = False
lowerT = 21.5
higherT = 24.5
lowerH = 83
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)

source = 'D4'
ignoreT = False
ignoreH = False
T2 = True
H2 = False
lowerT = 22
higherT = 26
lowerH = 85
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)


source = 'D3'
ignoreT = False
ignoreH = True
T2 = True
H2 = False
lowerT = 22
higherT = 24
lowerH = None
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)


source = 'D5'
ignoreT = False
ignoreH = False
T2 = True
H2 = True
lowerT = 22
higherT = 24.5
lowerH = 77
higherH = 85
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)


source = 'D7'
ignoreT = False
ignoreH = False
T2 = False
H2 = False
lowerT = 25
higherT = None
lowerH = 80
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
```

```
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)
```

```
Analysing for Dharwad
Disease type: D1
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 31.590651412079982, SE 19.50886987470224, Len 7
X2 (Mean of users checking Disease in un-fav condition) 6.515126050420168, SE 6.330623182247867, Len 15
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 6 is : 1.4397557472577691
T Statistics:3.3200927651509353 is gt than T Critical: 1.4397557472577691 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D2
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 40.134920634920626, SE 33.565536651346264, Len 5
X2 (Mean of users checking Disease in un-fav condition) 6.096486366382561, SE 7.463629544788599, Len 17
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 4 is : 1.5332062737131438
T Statistics:2.251261246997875 is gt than T Critical: 1.5332062737131438 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D4
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 39.166666666666664, SE 45.468242885679125, Len 3
X2 (Mean of users checking Disease in un-fav condition) 12.108752272838958, SE 12.79395896519898, Len 19
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 2 is : 1.88561808316415
T Statistics:1.0243513460539664 is less than T Critical: 1.88561808316415 at alpha(0.1)
Hence we can retain the NULL Hypothesis (ho)

Disease type: D3
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 40.269712430426715, SE 51.14705832340868, Len 14
X2 (Mean of users checking Disease in un-fav condition) 11.961659663865547, SE 16.825323356036495, Len 8
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 17 is : 1.3333793897216264
T Statistics:1.8988640725750916 is gt than T Critical: 1.3333793897216264 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D5
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 14.177489177489177, SE 14.069264069264067, Len 4
X2 (Mean of users checking Disease in un-fav condition) 13.067252827056748, SE 19.183808487079045, Len 18
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 5 is : 1.475884048782027
T Statistics:0.13276358065762542 is less than T Critical: 1.475884048782027 at alpha(0.1)
Hence we can retain the NULL Hypothesis (ho)

Disease type: D7
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 35.0, SE 21.213203435596427, Len 2
X2 (Mean of users checking Disease in un-fav condition) 19.908221925133688, SE 28.31825542990713, Len 20
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 1 is : 3.077683537207806
T Statistics:0.9269123652944348 is less than T Critical: 3.077683537207806 at alpha(0.1)
Hence we can retain the NULL Hypothesis (ho)
```

For Disease D1, D2, D3 for Dharwad, the claim that more people access the disease information during favourable periods hold true. For the rest of the Diseases it is not statistically significant

In [48]:
```
dharwad = pd.read_excel("./Jayalaxmi Agro Case Data .xlsx", sheet_name="Belagavi_weather")

print("Analysing for Belgavi")
source = 'D1'
ignoreT = False
ignoreH = False
T2 = True
H2 = False
lowerT = 20
higherT = 24
lowerH = 80
higherH = None
#dharwad.head()
alpha = 0.1

x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)

source = 'D2'
ignoreT = False
ignoreH = False
T2 = True
H2 = False
lowerT = 21.5
higherT = 24.5
lowerH = 83
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)

source = 'D4'
ignoreT = False
ignoreH = False
T2 = True
H2 = False
lowerT = 22
higherT = 26
lowerH = 85
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)


source = 'D3'
ignoreT = False
ignoreH = True
T2 = True
H2 = False
lowerT = 22
higherT = 24
lowerH = None
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)


source = 'D5'
ignoreT = False
ignoreH = False
T2 = True
H2 = True
lowerT = 22
higherT = 24.5
lowerH = 77
higherH = 85
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)


source = 'D7'
ignoreT = False
ignoreH = False
T2 = False
H2 = False
lowerT = 25
higherT = None
lowerH = 80
higherH = None
#dharwad.head()
alpha = 0.1
x1, S1, n1, x2, S2, n2 = prepareData(dharwad, source, ignoreT,
```

```
                ignoreH, T2, H2, lowerT, higherT, lowerH, higherH)

pairwiseT_test(alpha, x1, S1, n1, x2, S2, n2)
```

```
Analysing for Belgavi
Disease type: D1
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 37.593051847437806, SE 32.135791146151945, Len 9
X2 (Mean of users checking Disease in un-fav condition) 11.916694121951826, SE 13.214657338143653, Len 15
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 9 is : 1.3830287383964925
T Statistics:2.2839245979002736 is gt than T Critical: 1.3830287383964925 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D2
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 29.380223212460052, SE 14.257675698003663, Len 8
X2 (Mean of users checking Disease in un-fav condition) 9.173547458456326, SE 11.634011544592797, Len 16
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 11 is : 1.3634303180205214
T Statistics:3.4720833038193164 is gt than T Critical: 1.3634303180205214 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D4
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 24.28983784246942, SE 17.12396482492092, Len 5
X2 (Mean of users checking Disease in un-fav condition) 12.973835703960896, SE 11.293745993618758, Len 19
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 4 is : 1.5332062737131438
T Statistics:1.3997159471060217 is less than T Critical: 1.5332062737131438 at alpha(0.1)
Hence we can retain the NULL Hypothesis (ho)

Disease type: D3
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 30.957728757809885, SE 24.56035201007001, Len 13
X2 (Mean of users checking Disease in un-fav condition) 11.61232644458533, SE 16.414430640928842, Len 11
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 20 is : 1.325340706985046
T Statistics:2.297579720814241 is gt than T Critical: 1.325340706985046 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D5
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 36.574074074074076, SE 18.326406324471435, Len 4
X2 (Mean of users checking Disease in un-fav condition) 10.515473835553788, SE 11.909006454368914, Len 20
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 3 is : 1.6377443572159065
T Statistics:2.7308507082037217 is gt than T Critical: 1.6377443572159065 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)

Disease type: D7
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
X1 (Mean of users checking Disease in fav condition) 72.42328042328042, SE 15.394217855891272, Len 3
X2 (Mean of users checking Disease in un-fav condition) 21.006418135849568, SE 25.02226505813728, Len 21
Null hypothesis: The means of diseases access: x1 - x2 <= 0
Alternate hypothesis: x1 - x2 > 0
This is a right sided T-Test
Significant t-value at alpha - 0.1 and df: 3 is : 1.6377443572159065
T Statistics:4.929164561043593 is gt than T Critical: 1.6377443572159065 at alpha(0.1)
Hence we can reject the NULL Hypothesis (ho)
```

For Disease D1, D2, D3, D5, D7 for Belgavi, the claim that more people access the disease information during favourable periods hold true. For the rest of the Diseases it is not statistically significant

Q5 - 8

Conclusions:

- Farmers in Belgavi are more aware of Disease D5 and D7 and impact of weather on disease. They seem to be checking for disease in favourable weather conditions more than Dharwad Farmers
- Farmers are not cheking for D4 Diseases. May be they need to be made more aware of D4, favourable weather conditions for D4

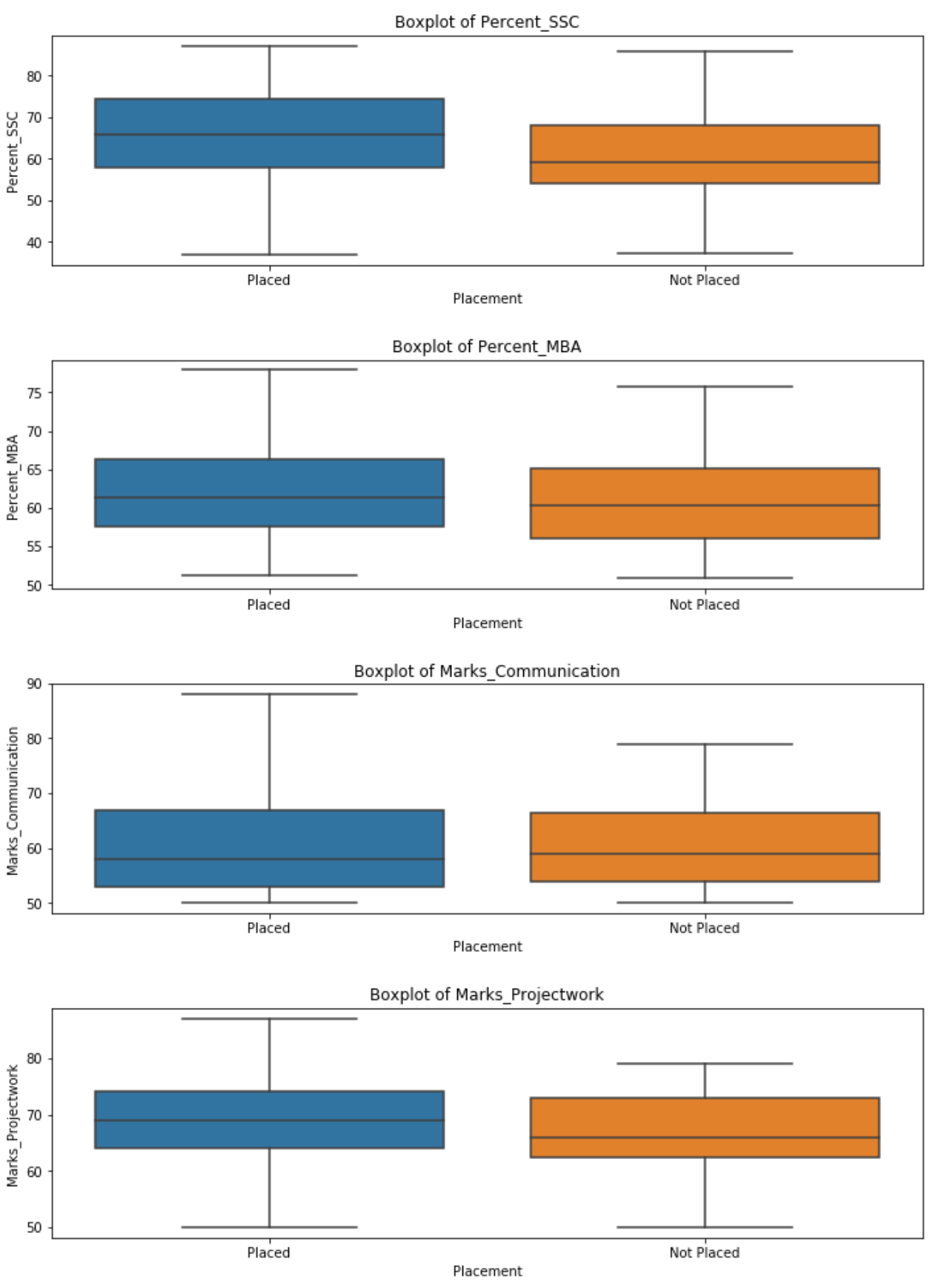
Q6

Q6 - 1

In [49]:
```
deandata = pd.read_excel("./IMB 485 Deans Dilemma Data.xlsx")
#deandata.info()
text = """
"The data has Nominal Variable (e.g. Gender, Board_SSC, Board_HSC etc.) ,
Continuous variables (e.g. Marks_Projectwork, Marks_Communication)
and Binary / indicator variables (e.g. Placement_B, Board_CBSE etc.)"
"""
print(text)
```

```
"The data has Nominal Variable (e.g. Gender, Board_SSC, Board_HSC etc.) ,
Continuous variables (e.g. Marks_Projectwork, Marks_Communication)
and Binary / indicator variables (e.g. Placement_B, Board_CBSE etc.)"
```

In [50]:
```
plt.figure(figsize = (10, 20))

for i, source in enumerate(['Percent_SSC', 'Percent_MBA',
                            'Marks_Communication','Marks_Projectwork']):
    plt.subplot(6, 1, i + 1)
    sns.boxplot(x = 'Placement', y=source, data= deandata)

    # Label the plots
    plt.title('Boxplot of %s' % source)

plt.tight_layout(h_pad = 2.5)
```

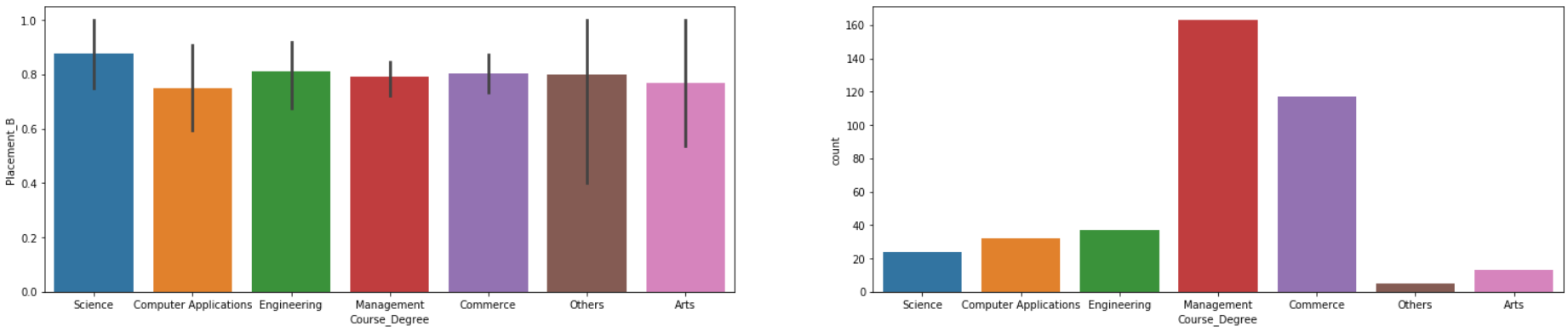


Observations

- SSC Median scores for students with placement is greater than students without placement
- Students who perform better in project work seems to be placed more

In [51]:
```
plt.figure(figsize=(26,5))
plt.subplot(1,2,1)
sns.barplot(x = 'Course_Degree', y='Placement_B', data = deandata)
plt.subplot(1,2,2)
sns.countplot(x = 'Course_Degree', data = deandata)
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9871fd39b0>

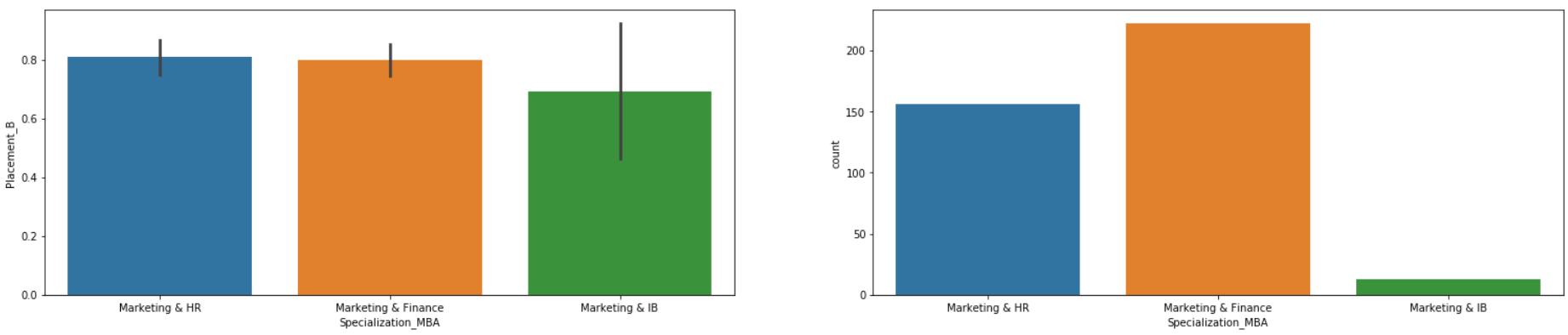


Observations

- In percentage terms, students from science background get getter placement than others however the number of students are too small to conclude anything from graphs. Proper Hypothesis test needs tobe performed to confirm statistical significance

In [52]:
```
plt.figure(figsize=(26,5))
plt.subplot(1,2,1)
sns.barplot(x = 'Specialization_MBA', y='Placement_B', data = deandata)
plt.subplot(1,2,2)
sns.countplot(x = 'Specialization_MBA', data = deandata)
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x7f98701001d0>



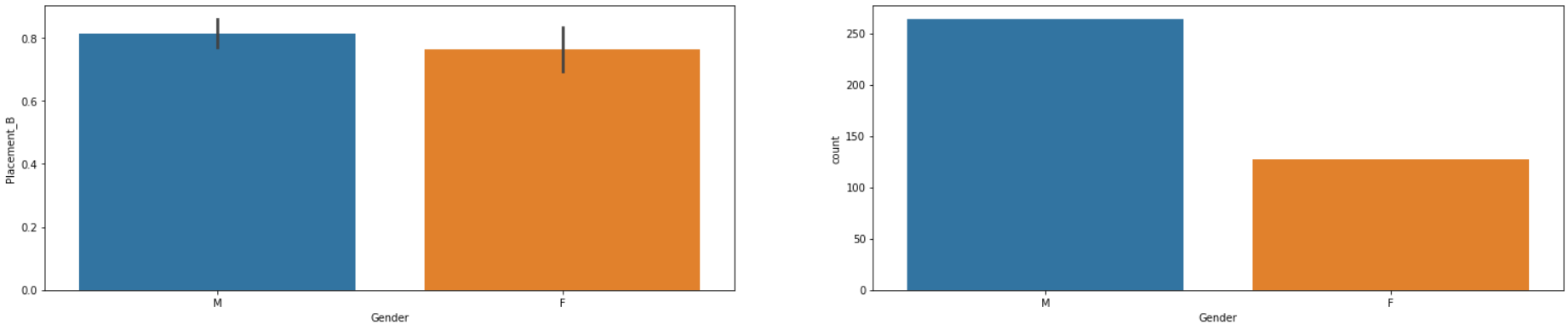
Observations

- Marketing & IB students seems to be doing worse than others in term of placements

However confirmation can only be provided after performing hypothesis tests

In [53]:
```
plt.figure(figsize=(26,5))
plt.subplot(1,2,1)
sns.barplot(x = 'Gender', y='Placement_B', data = deandata)
plt.subplot(1,2,2)
sns.countplot(x = 'Gender', data = deandata)
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x7f987010a0b8>



Observations

- There are significantly more Male students than Female students
- Percentage of Male students getting placed > percentage of Female students getting placed

In [54]:
```
datax = deandata[['Gender', 'Experience_Yrs','Board_SSC', 'Percent_SSC', 'Percent_HSC',
                  'Placement_B', 'Percent_MBA', 'Marks_Projectwork', 'Percentile_ET']].groupby(
    ['Board_SSC','Gender'], as_index = False).agg(['mean'])
datax.columns = ['_'.join(col).strip() for col in datax.columns.values]
datax
```

Out[54]:

Board_SSC	Gender	Experience_Yrs_mean	Percent_SSC_mean	Percent_HSC_mean	Placement_B_mean	Percent_MBA_mean	Marks_Projectwork_mean	Percentile_ET_m
CBSE	F	0.351351	63.135946	64.267568	0.810811	62.334054	68.972973	54.435
	M	0.526316	62.818421	63.512632	0.842105	60.106184	67.947368	58.351
ICSE	F	0.533333	68.935667	74.964333	0.800000	66.944333	71.966667	60.016
	M	0.574468	63.149574	64.671064	0.765957	60.172340	66.787234	54.668
Others	F	0.533333	69.083167	62.918167	0.716667	64.282833	70.566667	54.035
	M	0.418440	63.742908	61.528652	0.815603	60.597447	67.248227	52.614

Observations on Gender vs Placement

What we observe is that there is a trend of Female students getting less proportion of placement than Male students

- Except for students from CBSE board, Females have similar performance to males in all catgories SSC Marks, HSC Marks, ET Marks, and done well in MBA courses; but they end up landing with lesser jobs
- For CBSE Board performance of Females are similar to Males except on Years of Experience and ET Marks

For all the bins above the count of data points in each bracket is approximately >= 10% of the entire Data set, hence we will consider them to be decent sized buckets for analysis.

This observation needs to be statistically validated before prescribing corrective actions

In [55]:
```
datax = deandata.loc[deandata['Placement_B'] > 0, ['Gender','Board_SSC', 'Salary']].groupby(
    ['Board_SSC','Gender'], as_index = False).agg(['mean', len])
datax.columns = ['_'.join(col).strip() for col in datax.columns.values]
datax
```

Out[55]:

Board_SSC	Gender	Salary_mean	Salary_len
CBSE	F	228720.000000	30
	M	299328.125000	64
ICSE	F	275750.000000	24
	M	281805.555556	36
Others	F	257395.348837	43
	M	276608.695652	115

Observations on Gender vs Salary

For Students who were placed Females were offered lesser salary than Male counterparts.

This observation needs to be statistically validated before prescribing corrective actions

Q6 - 2

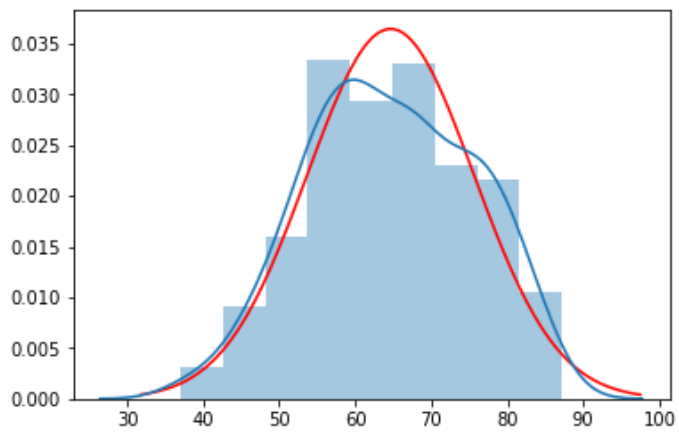
In [56]:
```
p = deandata.Placement_B.sum()/deandata.shape[0]
print("Proportion of students not placed: {}".format(1 - p))
print("Prob = comb(20,i) * (p) ** i * (1-p) ** (20 - i)")
from scipy.special import comb
print("Exactly 5 out of randomly selected 20 Students is: {}".format(comb(20,5) * (1-p) ** 5 * p ** 15))
print("\n")
x = []
for i in np.arange(0,6):
    x.append(round(comb(20,i) * ((1-p) ** i) * (p ** (20 - i)),4))
print("Prob of 0, 1, 2, 3, 4,5 students not been selected is (P): {}".format(x))
print("At least 5 out of randomly selected 20 Students is (1 - P): {}".format(1 - np.sum(x)))
```

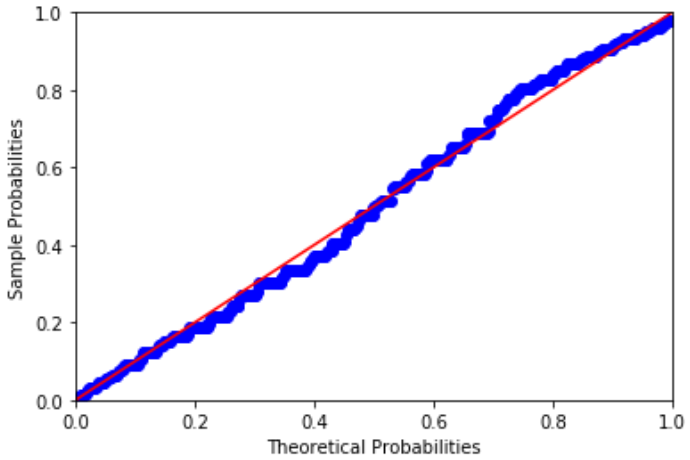
```
Proportion of students not placed: 0.20204603580562663
Prob = comb(20,i) * (p) ** i * (1-p) ** (20 - i)
Exactly 5 out of randomly selected 20 Students is: 0.17675144725366596


Prob of 0, 1, 2, 3, 4,5 students not been selected is (P): [0.011, 0.0555, 0.1334, 0.2027, 0.2181, 0.1768]
At least 5 out of randomly selected 20 Students is (1 - P): 0.20250000000000012
```

Q6 - 3 - a

The Optimal number of bins for histogram is computed using the following formula (this will be used later in the solution): **N = 1 =3.322 * Log10 (n)** where n = number of onservations

In [57]:
```
sslc = deandata[['Percent_SSC']]
#print(sslc.shape[0])

mu = deandata[['Percent_SSC']].values.mean()
sigma = deandata[['Percent_SSC']].values.std()
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
plt.plot(x,mlab.normpdf(x, mu, sigma), color = 'red')
sns.distplot(sslc, bins=9)
plt.show()
pp_x = sm.ProbPlot(sslc.values.reshape(-1), fit=True)
pp_x.ppplot(line='45')
plt.show()
```





In [58]:
```
%%R -i sslc
library(data.table)
options(warn=-1)

print(paste("ho = The distibutions are same (normal)"))
print(paste("ha = The distibutions are dissimilar"))

propUndVot = unlist(sslc$Percent_SSC)
#print(propUndVot[propUndVot >=76 & propUndVot <81.6])
Xmin = min(propUndVot)
print(paste("Min Range:", Xmin))
Xmax = max(propUndVot)
print(paste("Max Range:", Xmax))
N = 1 + 3.3*log10(length(propUndVot)) # Number of bins in the range
N = floor(N)
print(paste("Total Num Bins:", floor(N)))

obsdistribution = as.data.table(table(cut(propUndVot, breaks = seq(36.99, 87.21, by=((87.21 - 36.99)/N)))))#breaks = N)))
#print(obsdistribution)

cutpoint = unique(c(as.numeric( sub("\\((.+),.*", "\\1", obsdistribution$V1) ),
      as.numeric( sub("[^,]*,([^]]*)\\]", "\\1", obsdistribution$V1) )))
#print(cutpoint)

meandist = mean(propUndVot)
std = sd(propUndVot)
#print(meandist)
#print(std)

normaldist = pnorm(cutpoint, meandist , std)
#print(normaldist)
probval = c()
for(i in seq(1:length(normaldist)-1)){
    probval = c(probval, normaldist[i+1] - normaldist[i])
    #print(normaldist[i+1])
    #print(normaldist[i])
}
#print(probval)
normfreq = probval * length(propUndVot)


obsdistribution$ExpectedNorm = as.integer(normfreq[1:9])
obsdistribution$ExpectedNormDev = (obsdistribution$N - obsdistribution$ExpectedNorm)^2/
                                        ifelse(obsdistribution$ExpectedNorm==0, 0,
                                               obsdistribution$ExpectedNorm)

print(obsdistribution)
obsdistribution$ExpectedNormDev[is.infinite(obsdistribution$ExpectedNormDev)] = 0
#print(sum(obsdistribution$ExpectedNormDev))

print(paste0("Chisq:",sum(obsdistribution$ExpectedNormDev)))
print(paste("Distribution is Normal - Chisq Critical = 15.5, DF = ",(N-1)))
print("We retain the Null Hypothesis")
#install.packages("fitdistrplus")
#library(fitdistrplus)
#x = fitdist(propUndVot, "norm")
#summary(x)

#print(length(propUndVot))
```

```
[1] "ho = The distibutions are same (normal)"
[1] "ha = The distibutions are dissimilar"
[1] "Min Range: 37"
[1] "Max Range: 87.2"
[1] "Total Num Bins: 9"
           V1  N ExpectedNorm ExpectedNormDev
1:   (37,42.6]  7           6      0.16666667
2: (42.6,48.1] 20          16      1.00000000
3: (48.1,53.7] 35          36      0.02777778
4: (53.7,59.3] 73          60      2.81666667
5: (59.3,64.9] 64          76      1.89473684
6: (64.9,70.5] 72          75      0.12000000
7:   (70.5,76] 50          57      0.85964912
8:   (76,81.6] 47          34      4.97058824
9: (81.6,87.2] 23          16      3.06250000
[1] "Chisq:14.9185853113175"
[1] "Distribution is Normal - Chisq Critical = 15.5, DF =  8"
[1] "We retain the Null Hypothesis"
```

Q6 - 3 - b

First solution with SSC Marks == 60, and > 60. Since the problem is not clearly defined hence giving both options

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

S1, S2 are the Sample SD, n1, n2 are the sample lengths

X1 = Salary of Students with Avg SSC Marks == 60 / <= 60. X2 = Salary of Students with Avg SSC Marks >= 60.

The test tries to prove if the Salary for X1 is less than X2.

In [59]:

```
x1 = deandata.loc[(deandata['Percent_SSC'] == 60) & (deandata['Placement_B'] == 1), 'Salary'].mean()
S1 = deandata.loc[(deandata['Percent_SSC'] == 60) & (deandata['Placement_B'] == 1), 'Salary'].std()
n1 = deandata.loc[(deandata['Percent_SSC'] == 60)  & (deandata['Placement_B'] == 1), 'Salary'].shape[0]

print("X1 = Salary Stats for Percent == 60:: Mean: {}, STD: {}, Len: {}".format(x1, S1, n1))

x2 = deandata.loc[(deandata['Percent_SSC'] > 60) & (deandata['Placement_B'] == 1), 'Salary'].mean()
S2 = deandata.loc[(deandata['Percent_SSC'] > 60) & (deandata['Placement_B'] == 1), 'Salary'].std()
n2 = deandata.loc[(deandata['Percent_SSC'] > 60) & (deandata['Placement_B'] == 1), 'Salary'].shape[0]

print("X2 = Salary Stats for Percent > 60:: Mean: {}, STD: {}, Len: {}".format(x2, S2, n2))
ho = "For the means x1 - x2 >= 0"
ha = "x1 - x2 < 0"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a left sided T-Test")

print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SE {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("DF {}".format(df))

tstat = ((x1 - x2) - 0) /(Su)
print("T-stat {}".format(tstat))
alpha = 0.05

print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , sp.stats.t.ppf(alpha, df = df)))
print("p-value:{} is greater than alpha({})".format(1 - sp.stats.t.cdf(tstat, df = df), alpha))
print("Hence we can retain the NULL Hypothesis (ho)")
```

```
X1 = Salary Stats for Percent == 60:: Mean: 276230.76923076925, STD: 127706.14305646763, Len: 13
X2 = Salary Stats for Percent > 60:: Mean: 279859.40594059404, STD: 80594.30269880143, Len: 202
Null hypothesis: For the means x1 - x2 >= 0
Alternate hypothesis: x1 - x2 < 0
This is a left sided T-Test
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 35870.36750625153
DF 12
T-stat -0.1011597304987852
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : -1.7822875556491593
p-value:0.5394528767472087 is greater than alpha(0.05)
Hence we can retain the NULL Hypothesis (ho)
```

There is no statistical evidence to suggest that the average salary of students with average score of 60 marks in SLC is less than the average salary of students with an average score of more than 60 marks in SSLC

Q6 - 3 - b

Alternate solution with SSC Marks <= 60 and greater than 60. Since the problem is not clearly defined hence giving both options.

2 Sample T Test equations remain same

In [60]:
```
x1 = deandata.loc[(deandata['Percent_SSC'] <= 60) & (deandata['Placement_B'] == 1), 'Salary'].mean()
S1 = deandata.loc[(deandata['Percent_SSC'] <= 60) & (deandata['Placement_B'] == 1), 'Salary'].std()
n1 = deandata.loc[(deandata['Percent_SSC'] <= 60) & (deandata['Placement_B'] == 1), 'Salary'].shape[0]

print("X1 = Salary Stats for Percent <= 60:: Mean: {}, STD: {}, Len: {}".format(x1, S1, n1))

x2 = deandata.loc[(deandata['Percent_SSC'] > 60) & (deandata['Placement_B'] == 1), 'Salary'].mean()
S2 = deandata.loc[(deandata['Percent_SSC'] > 60) & (deandata['Placement_B'] == 1), 'Salary'].std()
n2 = deandata.loc[(deandata['Percent_SSC'] > 60) & (deandata['Placement_B'] == 1), 'Salary'].shape[0]

print("X2 = Salary Stats for Percent > 60:: Mean: {}, STD: {}, Len: {}".format(x2, S2, n2))
print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
ho = "For the means x1 - x2 >= 0"
ha = "x1 - x2 < 0"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a left sided T-Test")

Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SE {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("DF {}".format(df))

tstat = ((x1 - x2) - 0) /(Su)
print("T-Statistics: {}".format(tstat))

alpha = 0.05
print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , sp.stats.t.ppf(alpha, df = df)))
print("Hence we can retain the NULL Hypothesis (ho)")
```

```
X1 = Salary Stats for Percent <= 60:: Mean: 264800.0, STD: 112817.20214983808, Len: 110
X2 = Salary Stats for Percent > 60:: Mean: 279859.40594059404, STD: 80594.30269880143, Len: 202
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
Null hypothesis: For the means x1 - x2 >= 0
Alternate hypothesis: x1 - x2 < 0
This is a left sided T-Test
SE 12159.860487859336
DF 170
T-Statistics: -1.2384521973446707
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : -1.6538663174513013
Hence we can retain the NULL Hypothesis (ho)
```

There is no statistical evidence to suggest that the average salary of students with average score of 60 marks in SLC is less than the average salary of students with an average score of more than 60 marks in SSLC

Q6 - 3 - c

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

S1, S2 are the Sample SD, n1, n2 are the sample lengths

In [61]:
```
x1 = deandata.loc[(deandata['Gender'] == 'M') & (deandata['Placement_B'] == 1), 'Salary' ].mean()
S1 = deandata.loc[(deandata['Gender'] == 'M') & (deandata['Placement_B'] == 1), 'Salary'].std()
n1 = deandata.loc[(deandata['Gender'] == 'M') & (deandata['Placement_B'] == 1), 'Salary'].shape[0]

print("Stats for male: X1:Mean {}, STD {}, Len {}".format(x1, S1, n1))

x2 = deandata.loc[(deandata['Gender'] == 'F') & (deandata['Placement_B'] == 1), 'Salary'].mean()
S2 = deandata.loc[(deandata['Gender'] == 'F') & (deandata['Placement_B'] == 1), 'Salary'].std()
n2 = deandata.loc[(deandata['Gender'] == 'F') & (deandata['Placement_B'] == 1), 'Salary'].shape[0]

print("Stats for Female: X2:Mean {}, STD {}, Len {}".format(x2, S2, n2))
ho = "Null hypothesis: ho = x1 - x2 <= 10000"
ha = "Alternate Hypothesis: ha = x1 - x2 > 10000"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a right sided T-Test")

print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SU {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("Degrees of freedom: {}".format(df))

tstat = ((x1 - x2) - 10000) /(Su)
print("T-stat {}".format(tstat))

alpha = 0.05
print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , -1* sp.stats.t.ppf(alpha, df = df)))
print("p-value:{} is less than alpha({})".format(1 - sp.stats.t.cdf(tstat, df = df), alpha))
print("Hence we can reject the NULL Hypothesis (ho), Male Salary > Female Salary by 10K")
```

```
Stats for male: X1:Mean 284241.8604651163, STD 99430.42049537445, Len 215
Stats for Female: X2:Mean 253068.04123711342, STD 74190.54233637161, Len 97
Null hypothesis: Null hypothesis: ho = x1 - x2 <= 10000
Alternate hypothesis: Alternate Hypothesis: ha = x1 - x2 > 10000
This is a right sided T-Test
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SU 10135.482345249799
Degrees of freedom: 243
T-stat 2.08907859604002
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : 1.6511484017734768
p-value:0.018870644990801377 is less than alpha(0.05)
Hence we can reject the NULL Hypothesis (ho), Male Salary > Female Salary by 10K
```

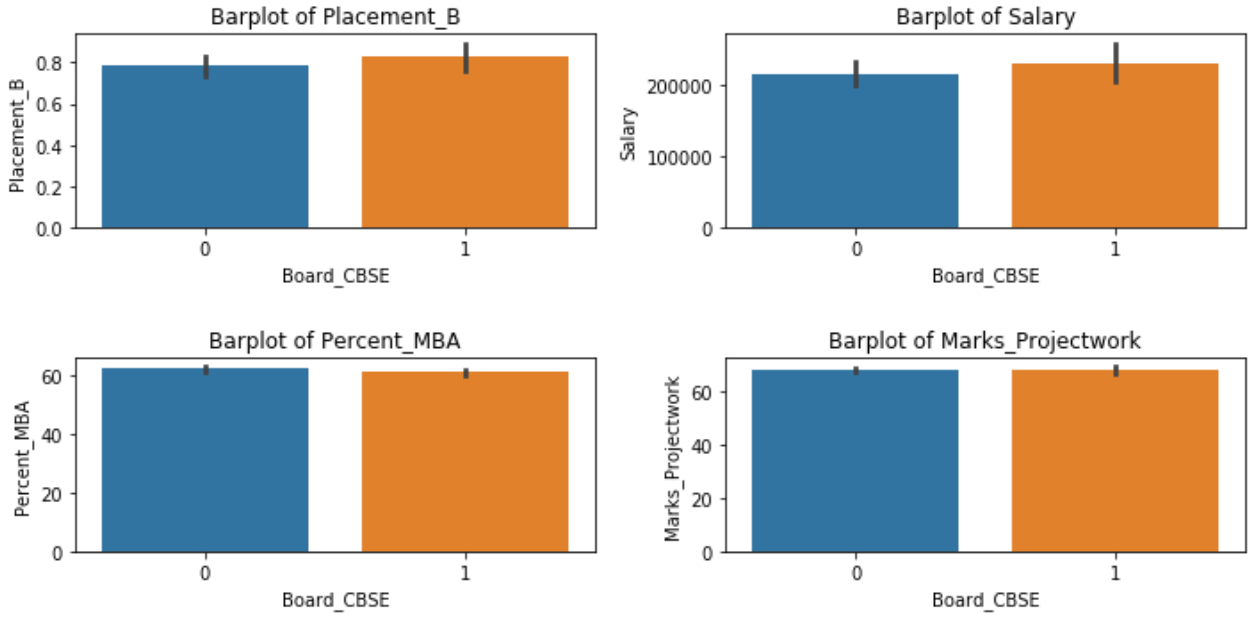
Q6 - 3 - d

Assumption is effectiveness is determined by the outcome variable Placement, i.e. how frequently CBSE students get placed verses placement of non-CBSE student. There are correlation of other KPIs vs CBSE / non-CBSE, even graphically we dont see any significant deviations. We will perform a T-Test only for CBSE and Placement subsequently after the plots

In [62]:
```
#deandata.loc[deandata["Board_CBSE"] == 1]
print("Correlation Board_CBSE and Placement\n", deandata[["Board_CBSE", "Placement_B"]].corr())
print("\nCorrelation Board_CBSE and Percent MBA Marks Obtained\n", deandata[["Board_CBSE", "Percent_MBA"]].corr())
```

```
Correlation Board_CBSE and Placement
              Board_CBSE  Placement_B
Board_CBSE     1.000000     0.053834
Placement_B    0.053834     1.000000

Correlation Board_CBSE and Percent MBA Marks Obtained
              Board_CBSE  Percent_MBA
Board_CBSE     1.000000    -0.090726
Percent_MBA   -0.090726     1.000000
```

In [63]:
```
plt.figure(figsize = (10, 5))

for i, source in enumerate(['Placement_B', 'Salary', 'Percent_MBA', 'Marks_Projectwork']):
    plt.subplot(2, 2, i + 1)
    sns.barplot(x = 'Board_CBSE', y=source, data = deandata)
    # Label the plots
    plt.title('Barplot of %s' % source)

plt.tight_layout(h_pad = 2.5)
```



From the above plots nothing conclusive is visible regarding CBSE students being better than their counterparts. We will perform hypothesis test now

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test for proportions.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

S1, S2 are the Sample SD, n1, n2 are the sample lengths

In [64]:
```
x1 = deandata.loc[deandata['Board_CBSE'] == 1, 'Placement_B'].mean()
S1 = deandata.loc[deandata['Board_CBSE'] == 1, 'Placement_B'].std()
n1 = deandata.loc[deandata['Board_CBSE'] == 1, 'Placement_B'].shape[0]

print("Stats for CBSE Placement: X1:Mean {}, STD {}, Len {}".format(x1, S1, n1))

x2 = deandata.loc[deandata['Board_CBSE'] == 0, 'Placement_B'].mean()
S2 = deandata.loc[deandata['Board_CBSE'] == 0, 'Placement_B'].std()
n2 = deandata.loc[deandata['Board_CBSE'] == 0, 'Placement_B'].shape[0]

print("Stats for NonCBSE Placement: X2:Mean {}, STD {}, Len {}".format(x2, S2, n2))

print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SE {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("Degrees of freedom: {}".format(df))

tstat = ((x1 - x2) - 0) /(Su)
print("T-stat {}".format(tstat))

ho = "Null Hypothesis: ho = x1 - x2 <= 0"
ha = "Alternate hypothesis: ha = x1 - x2 > 0"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a right sided T-Test")

print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , -1* sp.stats.t.ppf(alpha, df = df)))
print("p-value:{} is gt than alpha({})".format(1 - sp.stats.t.cdf(tstat, df = df), alpha))
print("Hence we can retain the NULL Hypothesis (ho)")

print("Conclusions:")
print("1. There is no statistical proof that CBSE students get more jobs")
print("2. They do not get more marks in MBA as indicated on the graphs (not statistically validated)")
print("Hence giving more preference to CBSE students adds no extra value to the institution")
```

```
Stats for CBSE Placement: X1:Mean 0.831858407079646, STD 0.37565787215935875, Len 113
Stats for NonCBSE Placement: X2:Mean 0.7841726618705036, STD 0.4121369848339013, Len 278
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 0.04312580767108984
Degrees of freedom: 226
T-stat 1.1057357017596072
Null hypothesis: Null Hypothesis: ho = x1 - x2 <= 0
Alternate hypothesis: Alternate hypothesis: ha = x1 - x2 > 0
This is a right sided T-Test
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : 1.651623859318793
p-value:0.1350083082855783 is gt than alpha(0.05)
Hence we can retain the NULL Hypothesis (ho)
Conclusions:
1. There is no statistical proof that CBSE students get more jobs
2. They do not get more marks in MBA as indicated on the graphs (not statistically validated)
Hence giving more preference to CBSE students adds no extra value to the institution
```

Q6 - 3 - e

Recommendations:

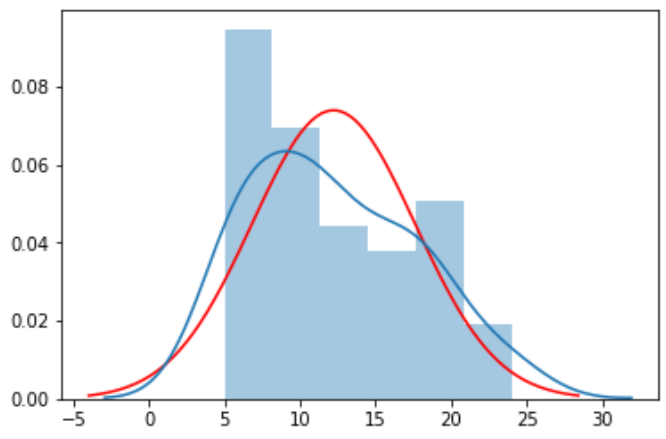
- Giving priority to CBSE students should be discontinued as there is no proof thet they get better placement or does better in Project Work / obtains overall higher percetage of marks
- More emphasis should be laid to doing Project Work well. It seems people with better Project Work, gets placed better (this has not been statistically proved, but graphs provide some visual evidence)
- Marketing students seems to be faring worse than the other streams in terms of placements
- Male students are paid significantly more than Female students on an average. There is a possibility of bias and gender pay gap (more tests are needed to provide complete evidence to this thought, effect of other variables like experience etc.). Hence he might need to work with recruiters on this if further studies do provide evidence to gender pay bias

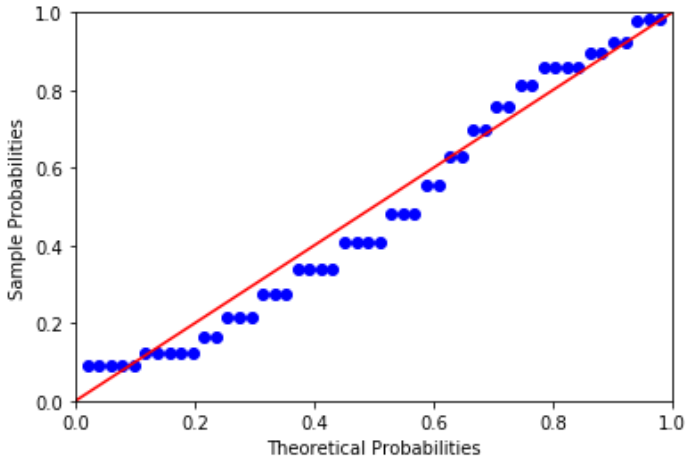
Q7

Q7 - a

The Optimal number of bins for histogram is computed using the following formula (this will be used later in the solution): **N = 1 =3.3 * Log10 (n)** where n = number of onservations

In [65]:
```
propUndVot = [12, 16, 12, 10, 14, 9, 8, 13, 5, 5,
19, 8, 6, 11, 19, 14, 10, 20, 11, 10,
6, 6, 5, 12, 16, 9, 5, 9, 17, 18,
15, 17, 18, 13, 18, 11, 7, 20, 6, 11,
23, 10, 24, 6, 24, 18, 7, 8, 5, 15]
mu = np.mean(propUndVot)
sigma = np.std(propUndVot)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
plt.plot(x,mlab.normpdf(x, mu, sigma), color = 'red')
sns.distplot(propUndVot, bins=6)
plt.show()
pp_x = sm.ProbPlot(np.array(propUndVot), fit=True)
pp_x.ppplot(line='45')
plt.show()
```





In [66]:
```
%%R -i propUndVot
print(paste("ho = The distibutions are same (normal)"))
print(paste("ha = The distibutions are dissimilar"))

propUndVot = unlist(propUndVot)
library(data.table)
Xmin = min(propUndVot)
print(paste("Min Range:", Xmin))
Xmax = max(propUndVot)
print(paste("Max Range:", Xmax))
N = 1 + 3.3*log10(length(propUndVot)) # Number of bins in the range
N = floor(N)
print(paste("Total Num Bins:", floor(N)))

obsdistribution = as.data.table(table(cut(propUndVot, breaks = seq(4.99,24.01, by = ((24.01-4.99)/N)))))#breaks = N)))
#print(obsdistribution)

cutpoint = unique(c(as.numeric( sub("\\((.+),.*", "\\1", obsdistribution$V1) ),
      as.numeric( sub("[^,]*,([^]]*)\\]", "\\1", obsdistribution$V1) )))
#print(cutpoint)

meandist = mean(propUndVot)
std = sd(propUndVot)
#print(meandist)
#print(std)

normaldist = pnorm(cutpoint, meandist , std)
#print(normaldist)
probval = c()
for(i in seq(1:length(normaldist)-1)){
    probval = c(probval, normaldist[i+1] - normaldist[i])
    #print(normaldist[i+1])
    #print(normaldist[i])
}

normfreq = probval * length(propUndVot)


obsdistribution$ExpectedNorm = as.integer(normfreq[1:6])
obsdistribution$ExpectedNormDev = (obsdistribution$N - obsdistribution$ExpectedNorm)^2/
                                        ifelse(obsdistribution$ExpectedNorm==0, 0,
                                               obsdistribution$ExpectedNorm)

print(obsdistribution)
obsdistribution$ExpectedNormDev[is.infinite(obsdistribution$ExpectedNormDev)] = 0
print(paste0("ChiSq:",(sum(obsdistribution$ExpectedNormDev)- 0.5)))
print("We have rejected the last bucket as the number of records in the bin is < 5")

print("Not Normal - Chisq Critical = 9.2, DF = 5")
print("We reject the Null Hypothesis, the distribution is statistically not normal")

#install.packages("fitdistrplus")
#library(fitdistrplus)
#x = fitdist(propUndVot, "norm")
#summary(x)
```

```
[1] "ho = The distibutions are same (normal)"
[1] "ha = The distibutions are dissimilar"
[1] "Min Range: 5"
[1] "Max Range: 24"
[1] "Total Num Bins: 6"
           V1  N ExpectedNorm ExpectedNormDev
1: (4.99,8.16] 15            6       13.500000
2: (8.16,11.3] 11           10        0.100000
3: (11.3,14.5]  7           11        1.454545
4: (14.5,17.7]  6            9        1.000000
5: (17.7,20.8]  8            4        4.000000
6:   (20.8,24]  3            2        0.500000
[1] "ChiSq:20.0545454545455"
[1] "We have rejected the last bucket as the number of records in the bin is < 5"
[1] "Not Normal - Chisq Critical = 9.2, DF = 5"
[1] "We reject the Null Hypothesis, the distribution is statistically not normal"
```

Q7 - b

We will calculate the lower and upper bound of CI using the following equation:

$$X \pm T * (\frac{S}{\sqrt{n}})$$

where X = mean of sample T = Critical T-statistic Value at alpha/significance S = Standard Deviation

In [67]:
```
print("T distribution with 90% CI (Population variance in unknown)")
n = len(propUndVot)
alpha = 0.1
T = abs(sp.stats.t.ppf(alpha / 2, df = n - 1))
print("T-stat {} at alpha {}".format(T, alpha))
Xbar = np.mean(propUndVot)
S = np.std(propUndVot)

print("X_bar {}".format(Xbar))
print("SE {}".format(S))
lowerBound = Xbar - T * S / n ** 0.5
print("lowerBound {}".format(lowerBound))
upperBound = Xbar + T * S / n ** 0.5
print("upperBound {}".format(upperBound))
```

```
T distribution with 90% CI (Population variance in unknown)
T-stat 1.6765508919142635 at alpha 0.1
X_bar 12.22
SE 5.397369729785055
lowerBound 10.940283092282368
upperBound 13.499716907717634
```

Q8

We will use paired 2 sample T-test.

T-statistic equation:

$$T = \frac{D - \mu_d}{\frac{S_d}{\sqrt{n}}}$$

where Sd = SD

Here we have

$$\mu_d = 0$$

In [68]:
```
students = pd.DataFrame({"pair_num": np.arange(1,11),
                         "meditation": [4.0, 2.65, 3.65, 2.55, 3.2, 3.6, 2.9, 3.41, 3.33, 2.90],
                         "no_meditation": [3.75, 2.75, 3.45, 2.11, 3.21, 3.25, 2.58, 3.28, 3.35, 2.65]
                        })
students["D"] = students.meditation - students.no_meditation
students
```

Out[68]:

	meditation	no_meditation	pair_num	D
0	4.00	3.75	1	0.25
1	2.65	2.75	2	-0.10
2	3.65	3.45	3	0.20
3	2.55	2.11	4	0.44
4	3.20	3.21	5	-0.01
5	3.60	3.25	6	0.35
6	2.90	2.58	7	0.32
7	3.41	3.28	8	0.13
8	3.33	3.35	9	-0.02
9	2.90	2.65	10	0.25

In [69]:
```
print("We will use Paired Sample T test")
ho = "Difference in CGPA between students performing meditation and not performing meditation <= 0"
ha = "Difference in CGPA between students performing meditation and not performing meditation > 0"
alpha =0.05

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a right sided T-Test")

dmean = students.D.mean()
dstd = students.D.std()
print("Mean of difference: {}".format(dmean))
print("SD of difference: {}".format(dstd))
print("Null Hypothesis: ho = mean <= 0")
print("Alternate Hypothesis: ha = mean > 0")

print("Tstat Formula : tsat = (dmean - 0) / dstd / n ** 0.5")
tstat = (dmean - 0) / (dstd / students.shape[0] ** 0.5)
print("T-Statistic Value: {}".format(tstat))

print("For Alpha {} t-Critical is : {}".format(alpha, -1* sp.stats.t.ppf(alpha, df = students.shape[0] - 1)))
print("We reject null hypothesis ho. There is statistical evidence that meditation helps")
```

```
We will use Paired Sample T test
Null hypothesis: Difference in CGPA between students performing meditation and not performing meditation <= 0
Alternate hypothesis: Difference in CGPA between students performing meditation and not performing meditation > 0
This is a right sided T-Test
Mean of difference: 0.181
SD of difference: 0.17741664709566196
Null Hypothesis: ho = mean <= 0
Alternate Hypothesis: ha = mean > 0
Tstat Formula : tsat = (dmean - 0) / dstd / n ** 0.5
T-Statistic Value: 3.2261474098417446
For Alpha 0.05 t-Critical is : 1.8331129326536337
We reject null hypothesis ho. There is statistical evidence that meditation helps
```

Q9

We will use 2 sample T-test (with unknown SD and unequal SD) for this hypothesis test.

T-statistic equation:

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1}{n_1} + \frac{S_2}{n_2}}}$$

DF equation :

$$df = \left\lfloor \frac{S_u^4}{\frac{(\frac{S_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2 - 1}} \right\rfloor$$

SD (Su):

$$S_u = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

S1, S2 are the Sample SD, n1, n2 are the sample lengths

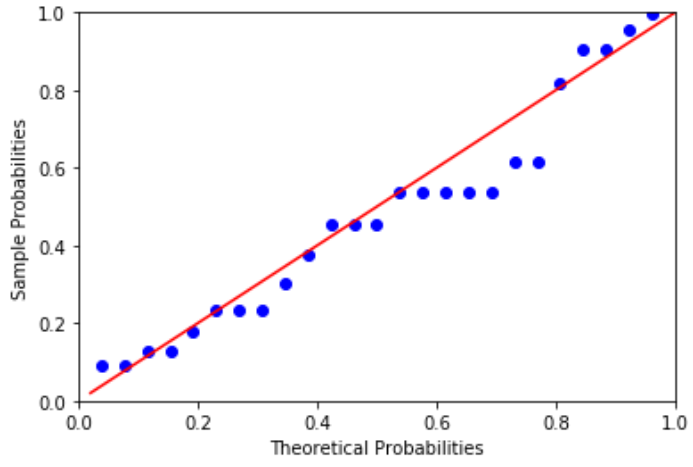
In [70]:
```
emergingMkt = [
    11.20, 12.10, 13.33, 16.40, 15.00, 10.00, 12.00, 13.00, 12.00, 13.00,
    8.25, 7.00, 10.00, 11.46, 11.00, 7.70, 7.00, 12.00, 18.00, 10.00,
    13.11, 9.00, 14.00, 9.90, 16.00, 9.00, 6.00, 11.40, 7.00, 16.00,
    8.41, 17.21, 14.00, 15.00, 17.20, 18.00, 9.00, 7.00, 15.45, 15.00,
    13.00, 18.60, 16.00, 9.60, 12.00, 6.00, 15.00, 8.00, 16.29, 9.00]
derivativeMkt = [
    17.65, 10.20, 19.00, 14.00, 11.00, 4.97, 11.00, 7.00, 5.12, 4.90,
    19.00, 11.45, 16.00, 6.87, 14.00, 8.00, 10.78, 16.00, 18.00, 11.00,
    13.00, 17.00, 18.00, 16.00, 12.00, 13.26, 19.00, 10.00, 17.00, 5.56,
    8.00, 15.55, 11.22, 6.78, 10.00, 19.00, 14.00, 15.00, 14.00, 7.00,
    14.00, 15.00, 18.00, 7.78, 10.00, 15.00, 16.20, 15.00, 11.65, 13.00
]

print("Emerging Market: Mean (x1): {}, SD: {}".format(np.mean(emergingMkt), np.std(emergingMkt)))
print("Derivative Market: Mean (x2): {}, SD: {}".format(np.mean(derivativeMkt), np.std(derivativeMkt)))

print("T-test with unknown population SD and unequal variance will be used for hypothesis testing")

n1 = len(emergingMkt)
x1 = np.mean(emergingMkt)
S1 = np.std(emergingMkt)

n2 = len(derivativeMkt)
x2 = np.mean(derivativeMkt)
S2 = np.std(derivativeMkt)


print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
print("SE {}".format(Su))

df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
print("DF {}".format(df))

tstat = ((x1 - x2) - 0) /(Su)
print("T-stat {}".format(tstat))

ho = "Null Hypothesis: ho = x1 - x2 <= 0"
ha = "Alternate hypothesis: ha = x1 - x2 > 0"

print("Null hypothesis: {}".format(ho))
print("Alternate hypothesis: {}".format(ha))
print("This is a right sided T-Test")

print("alpha/ Significance: {}".format(alpha))
print("Significant t-value at alpha - {} is : {}".format(alpha , -1*sp.stats.t.ppf(alpha, df = df)))
print("p-value:{} is greater than alpha({})".format(sp.stats.t.cdf(tstat, df = df), alpha))
print("Hence we can retain the NULL Hypothesis (ho)")
```

```
Emerging Market: Mean (x1): 12.0322, SD: 3.4964898341050556
Derivative Market: Mean (x2): 12.6588, SD: 4.158908818428219
T-test with unknown population SD and unequal variance will be used for hypothesis testing
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 0.7684004648619104
DF 95
T-stat -0.8154602042212539
Null hypothesis: Null Hypothesis: ho = x1 - x2 <= 0
Alternate hypothesis: Alternate hypothesis: ha = x1 - x2 > 0
This is a right sided T-Test
alpha/ Significance: 0.05
Significant t-value at alpha - 0.05 is : 1.6610518172519093
p-value:0.2084243682887848 is greater than alpha(0.05)
Hence we can retain the NULL Hypothesis (ho)
```

There is no statistical evidence that average returns of the hedge fund ‘Emerging Market’ is higher than that of ‘Derivative’.

Q10

In [71]:
```
diet = [0, 4, 3, 5, -3, 10, 0, 4, -2]
exercise = [-3, -1, 8,4 ,2 ,3]
modificbeh = [10, 1, 0, 12, 18, 4, -2, 5, 3, 4]

pp_x = sm.ProbPlot(np.array(diet + exercise + modificbeh), fit=True)
pp_x.ppplot(line='45')
plt.show()

print("PP Plot may not be normal by visual inspection. Morreover sample size is small.\n")
```




```
PP Plot may not be normal by visual inspection. Morreover sample size is small.
```

In [72]:
```
print("Diet == Mean (x1): {}, SD: {}, Count: {}".format(np.mean(diet), np.std(diet), len(diet)))
print("Exer == Mean (x2): {}, SD: {}, Count: {}".format(np.mean(exercise), np.std(exercise), len(exercise)))
print("ModB == Mean (x3): {}, SD: {}, Count: {}".format(np.mean(modificbeh), np.std(modificbeh), len(modificbeh)))

print("We cannot use Anova as the variances are not similar, we will use multiple pairwise t tests.",
      "We will use 3 pairwise tests. Since this increases the Type 1 error, we will use Bonferroni's Correction")


x1, S1, n1 = np.mean(diet), np.std(diet), len(diet)
x2, S2, n2 = np.mean(exercise), np.std(exercise), len(exercise)
x3, S3, n3 = np.mean(modificbeh), np.std(modificbeh), len(modificbeh)

alpha = 0.05
adjustedAlpha = alpha / 3

print("Adjusted Alpha: {}".format(adjustedAlpha))
ho = "x1 - x2 <= 0"
ha = "x1 - x2 >0"

def pairwise_t_test(S1, S2, n1, n2, x1, x2, adjustedAlpha):
    print("NUll Hypothesis: {}".format(ho))
    print("Alternate Hypothesis: {}".format(ha))
    print("This is 2 Sample T test, with unknown population SD and the SD of the two are unequal")
    Su = ((S1 ** 2) / n1 + (S2 ** 2) / n2) ** 0.5
    print("SE {}".format(Su))

    df = np.math.floor(Su ** 4 / ((((S1 ** 2) / n1) ** 2) / (n1 -1) + (((S2 ** 2) / n2) ** 2) / (n2 -1)))
    print("DF {}".format(df))

    tstat = ((x1 - x2) - 0) /(Su)
    print("T-stat {}".format(tstat))

    print("This is a two sided T-Test")

    #print("alpha/ Significance: {}".format(adjustedAlpha / 2))
    print("Significant t-value at alpha - {} is : {}".format(adjustedAlpha , -1*sp.stats.t.ppf(adjustedAlpha,
                                                                                              df = df)))
    print("p-value:{} is greater than alpha({})".format(1 - sp.stats.t.cdf(tstat, df = df), adjustedAlpha))
    print("Hence we can retain the NULL Hypothesis (ho)")
    print("\n\n")

pairwise_t_test(S1, S2, n1, n2, x1, x2, adjustedAlpha)

ho = "x2 - x3 <= 0"
ha = "x2 - x3 >0"
pairwise_t_test(S2, S3, n2, n3, x2, x3, adjustedAlpha)

ho = "x1 - x3 <= 0"
ha = "x1 - x3 >0"
pairwise_t_test(S1, S3, n1, n3, x1, x3, adjustedAlpha)

print("There is not enough statistically significant difference in weight loss between three weight loss programs")
```

```
Diet == Mean (x1): 2.3333333333333335, SD: 3.80058475033046, Count: 9
Exer == Mean (x2): 2.1666666666666665, SD: 3.531603350069515, Count: 6
ModB == Mean (x3): 5.5, SD: 5.80086200490927, Count: 10
We cannot use Anova as the variances are not similar, we will use multiple pairwise t tests. We will use 3 pairwise tests.
Since this increases the Type 1 error, we will use Bonferroni's Correction
Adjusted Alpha: 0.016666666666666666
NUll Hypothesis: x1 - x2 <= 0
Alternate Hypothesis: x1 - x2 >0
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 1.9192816300138555
DF 11
T-stat 0.08683804610033379
This is a two sided T-Test
Significant t-value at alpha - 0.016666666666666666 is : 2.431291192825905
p-value:0.4661804170878926 is greater than alpha(0.016666666666666666)
Hence we can retain the NULL Hypothesis (ho)



NUll Hypothesis: x2 - x3 <= 0
Alternate Hypothesis: x2 - x3 >0
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 2.3331745977752507
DF 13
T-stat -1.428668620218891
This is a two sided T-Test
Significant t-value at alpha - 0.016666666666666666 is : 2.3795918574465182
p-value:0.9116578507983054 is greater than alpha(0.016666666666666666)
Hence we can retain the NULL Hypothesis (ho)



NUll Hypothesis: x1 - x3 <= 0
Alternate Hypothesis: x1 - x3 >0
This is 2 Sample T test, with unknown population SD and the SD of the two are unequal
SE 2.229335836433115
DF 15
T-stat -1.420452950567223
This is a two sided T-Test
Significant t-value at alpha - 0.016666666666666666 is : 2.342924621576576
p-value:0.9120359852133058 is greater than alpha(0.016666666666666666)
Hence we can retain the NULL Hypothesis (ho)



There is not enough statistically significant difference in weight loss between three weight loss programs
```

END

In [ ]: