In [41]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
import statsmodels
```

In [2]:
```
print('Numpy Version',np.__version__)
print('Pandas Version',pd.__version__)
print('Seaborn Version',sns.__version__)
print('Matplotlib Version',matplotlib.__version__)
print('Statsmodels Version',statsmodels.__version__)
```

```
Numpy Version 1.22.3
Pandas Version 1.4.2
Seaborn Version 0.11.2
Matplotlib Version 3.5.1
Statsmodels Version 0.13.2
```

In [42]:
```
# Importing the data:

expected_ctc= pd.read_csv('expected_ctc.csv')
expected_ctc.head(10)
```

Out[42]:

	IDX	Applicant_ID	Total_Experience	Total_Experience_in_field_applied	Department	
0	1	22753	0	0	NaN	
1	2	51087	23	14	HR	Cons
2	3	38413	21	12	Top Management	Cons
3	4	11501	15	8	Banking	Fina Ar
4	5	58941	10	5	Sales	P Ma
5	6	30564	16	3	Top Management	Area Ma
6	7	27267	1	1	Engineering	
7	8	36521	19	11	Others	Ar
8	9	11616	8	7	Analytics/BI	C
9	10	43886	15	15	Analytics/BI	

In [43]:
```
# Accomodating all columns on screen:

pd.options.display.max_columns = None
```

In [5]:
```
expected_ctc.head(10)
```

Out[5]:

	IDX	Applicant_ID	Total_Experience	Total_Experience_in_field_applied	Department	
0	1	22753	0	0	NaN	
1	2	51087	23	14	HR	Cons
2	3	38413	21	12	Top Management	Cons
3	4	11501	15	8	Banking	Fina Ar
4	5	58941	10	5	Sales	P Ma
5	6	30564	16	3	Top Management	Area Ma
6	7	27267	1	1	Engineering	
7	8	36521	19	11	Others	Ar
8	9	11616	8	7	Analytics/BI	C
9	10	43886	15	15	Analytics/BI	

In [6]:
```
expected_ctc.tail(10)
```

Out[6]:

	IDX	Applicant_ID	Total_Experience	Total_Experience_in_field_applied	Department
24990	24991	34589	22	1	Top Management
24991	24992	13280	1	1	Sales
24992	24993	35325	25	12	Sales
24993	24994	31883	15	13	Healthcare
24994	24995	32035	7	3	Top Management
24995	24996	25550	18	13	Engineering
24996	24997	53442	12	8	HR
24997	24998	15777	22	8	Banking
24998	24999	57616	25	8	Marketing
24999	25000	20788	8	0	Banking

In [44]:
```
# Checking the total entries:

expected_ctc.size
```

Out[44]: 725000

In [45]:
```
# Checking the data structure:

expected_ctc.shape
```

Out[45]: (25000, 29)

In [9]:
```
# Checking the data types:

expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 29 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   IDX                                   25000 non-null  int64  
 1   Applicant_ID                          25000 non-null  int64  
 2   Total_Experience                      25000 non-null  int64  
 3   Total_Experience_in_field_applied     25000 non-null  int64  
 4   Department                            22222 non-null  object 
 5   Role                                  24037 non-null  object 
 6   Industry                              24092 non-null  object 
 7   Organization                          24092 non-null  object 
 8   Designation                           21871 non-null  object 
 9   Education                             25000 non-null  object 
 10  Graduation_Specialization             18820 non-null  object 
 11  University_Grad                       18820 non-null  object 
 12  Passing_Year_Of_Graduation            18820 non-null  float64
 13  PG_Specialization                     17308 non-null  object 
 14  University_PG                         17308 non-null  object 
 15  Passing_Year_Of_PG                    17308 non-null  float64
 16  PHD_Specialization                    13119 non-null  object 
 17  University_PHD                        13119 non-null  object 
 18  Passing_Year_Of_PHD                   13119 non-null  float64
 19  Curent_Location                       25000 non-null  object 
 20  Preferred_location                    25000 non-null  object 
 21  Current_CTC                           25000 non-null  int64  
 22  Inhand_Offer                          25000 non-null  object 
 23  Last_Appraisal_Rating                 24092 non-null  object 
 24  No_Of_Companies_worked                25000 non-null  int64  
 25  Number_of_Publications                25000 non-null  int64  
 26  Certifications                        25000 non-null  int64  
 27  International_degree_any              25000 non-null  int64  
 28  Expected_CTC                          25000 non-null  int64  
dtypes: float64(3), int64(10), object(16)
memory usage: 5.5+ MB
```

In [59]:
```
#Checking for duplicate values

expected_ctc.duplicated().sum()
```

Out[59]: 0

In [11]:
```
#There seem to be many features with missing values. Getting a coun

expected_ctc.isna().sum().sort_values(ascending=False)
```

Out[11]:
```
Passing_Year_Of_PHD                11881
University_PHD                     11881
PHD_Specialization                 11881
University_PG                       7692
Passing_Year_Of_PG                  7692
PG_Specialization                   7692
University_Grad                     6180
Passing_Year_Of_Graduation          6180
Graduation_Specialization           6180
Designation                         3129
Department                          2778
Role                                 963
Organization                         908
Industry                             908
Last_Appraisal_Rating                908
Number_of_Publications                 0
Current_CTC                            0
No_Of_Companies_worked                 0
Certifications                         0
International_degree_any               0
Inhand_Offer                           0
IDX                                    0
Preferred_location                     0
Curent_Location                        0
Applicant_ID                           0
Education                              0
Total_Experience_in_field_applied      0
Total_Experience                       0
Expected_CTC                           0
dtype: int64
```

In [46]:
```
#Dropping unnecessary columns:

expected_ctc=expected_ctc.drop(columns=["IDX","Applicant_ID"])
```

In [13]:
```
expected_ctc.head()
```

Out[13]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	Or
0	0	0	NaN	NaN	NaN	
1	23	14	HR	Consultant	Analytics	
2	21	12	Top Management	Consultant	Training	
3	15	8	Banking	Financial Analyst	Aviation	
4	10	5	Sales	Project Manager	Insurance	

In [47]:
```
expected_ctc.shape
```

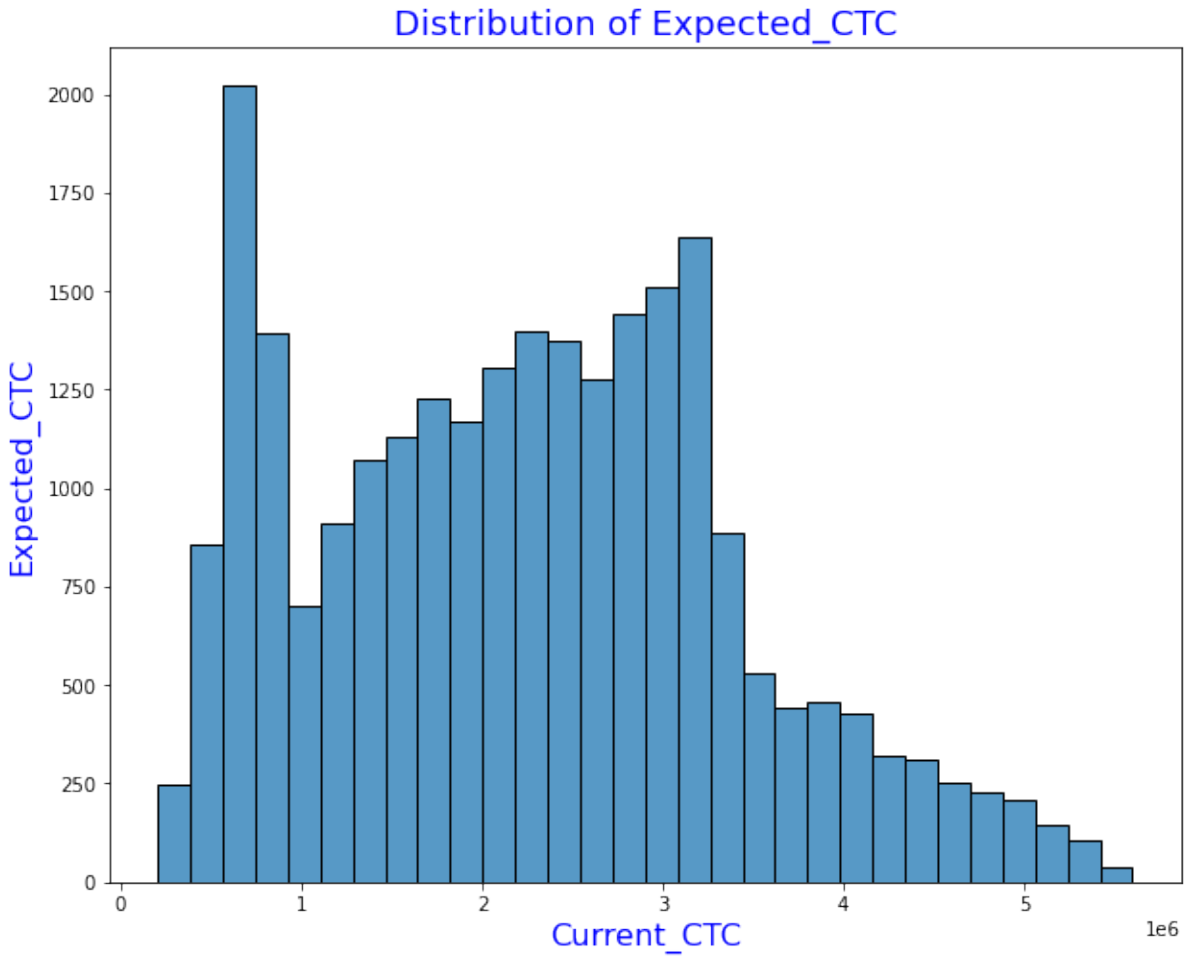
Out[47]: (25000, 27)

In [15]:
```
# Checking the distribution of target variable:

expected_ctc.describe()["Expected_CTC"]
```

Out[15]:
```
count    2.500000e+04
mean     2.250155e+06
std      1.160480e+06
min      2.037440e+05
25%      1.306278e+06
50%      2.252136e+06
75%      3.051354e+06
max      5.599570e+06
Name: Expected_CTC, dtype: float64
```

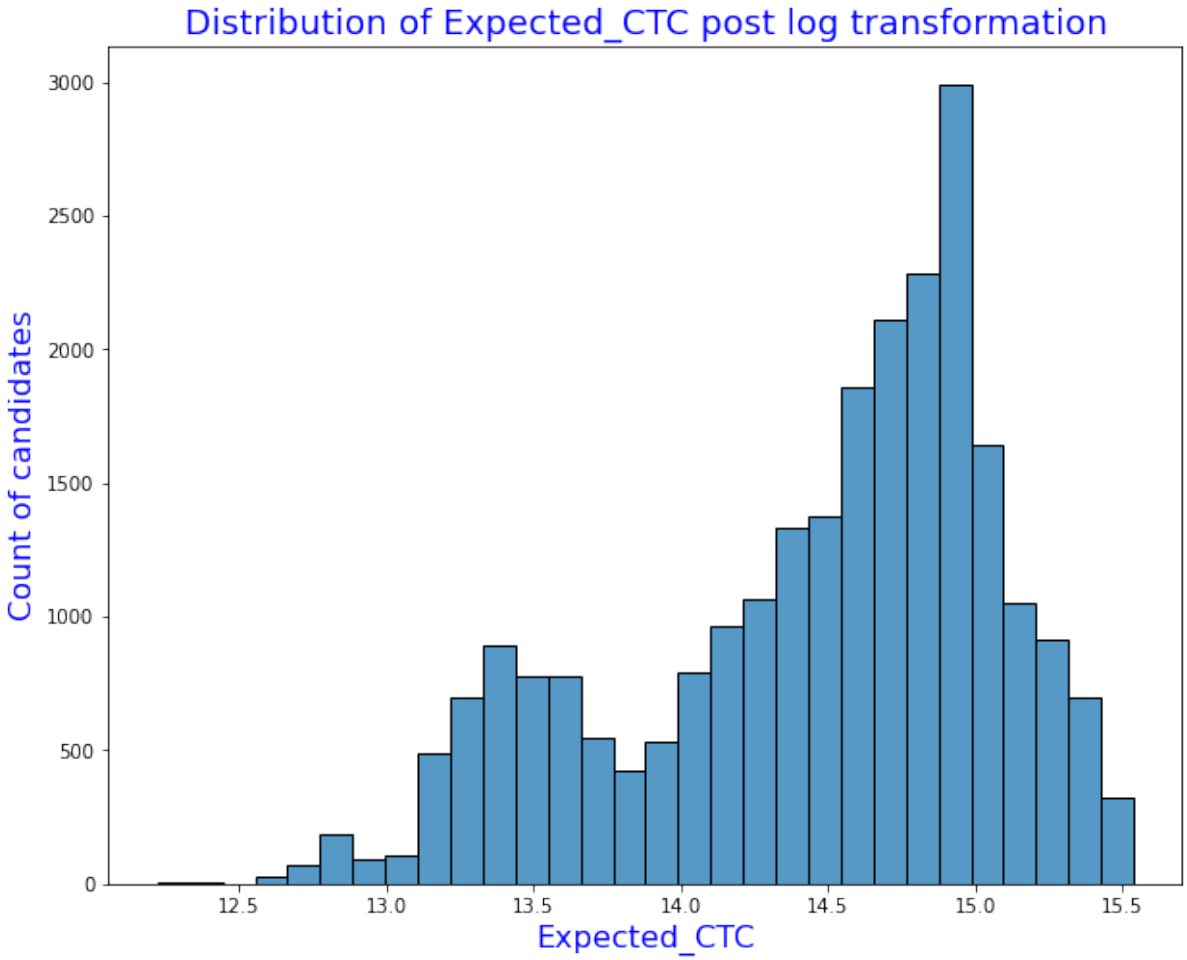
In [16]:
```
# Bin-wise distribution of target variable (30 bins):

plt.figure(figsize=(10,8))
sns.histplot(expected_ctc['Expected_CTC'],bins=30)
plt.labels=12
plt.title('Distribution of Expected_CTC',color='blue',fontsize=18);
plt.xlabel("Current_CTC", color="blue",size=16)
plt.ylabel("Expected_CTC",color="blue",size=16)
```

Out[16]:
```
Text(0, 0.5, 'Expected_CTC')
```



In [17]:
```
# Distribution of target variable post log transformation to normal

plt.figure(figsize=(10,8))
sns.histplot(np.log(expected_ctc['Expected_CTC']),bins=30)
plt.title('Distribution of Expected_CTC post log transformation',co
plt.xlabel("Expected_CTC", color="blue",size=16)
plt.ylabel("Count of candidates",color="blue",size=16)
```

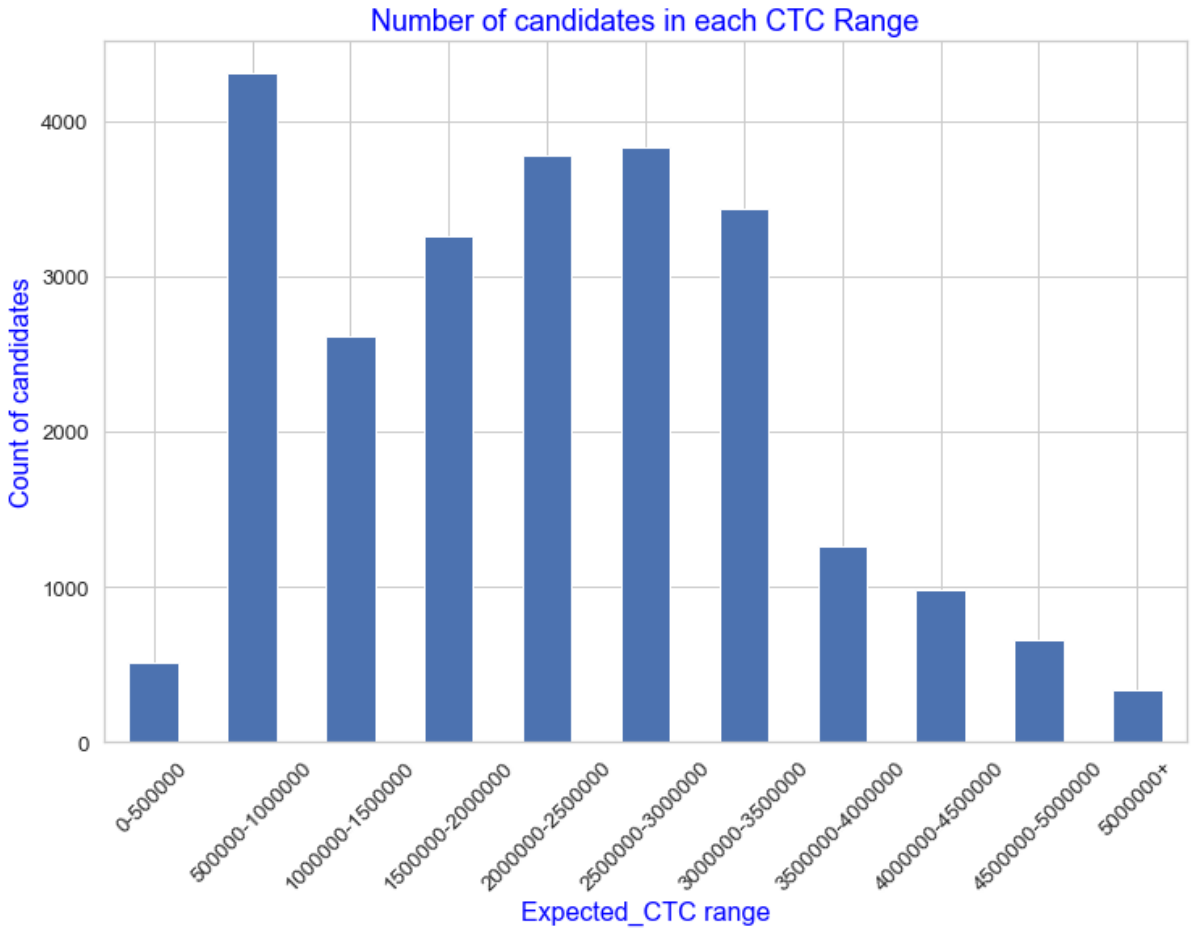
Out[17]: Text(0, 0.5, 'Count of candidates')



In [18]:
```
# Checking percentile distribution of target variable:

print(" 5% candidates have an expected_ctc lower than {0: .2f}".for
print("10% candidates have an expected_ctc lower than {0: .2f}".for
print("12.5% candidates have an expected_ctc lower than {0: .2f}".f
print("15% candidates have an expected_ctc lower than {0: .2f}".for
print("20% candidates have an expected_ctc lower than {0: .2f}".for
print("25% candidates have an expected_ctc lower than {0: .2f}".for
print("35% candidates have an expected_ctc lower than {0: .2f}".for
print("50% candidates have an expected_ctc lower than {0: .2f}".for
print("65% candidates have an expected_ctc lower than {0: .2f}".for
print("75% candidates have an expected_ctc lower than {0: .2f}".for
print("85% candidates have an expected_ctc lower than {0: .2f}".for
print("90% candidates have an expected_ctc lower than {0: .2f}".for
print("95% candidates have an expected_ctc lower than {0: .2f}".for
```

```
 5% candidates have an expected_ctc lower than  577473.50
10% candidates have an expected_ctc lower than  681085.10
12.5% candidates have an expected_ctc lower than  743618.38
15% candidates have an expected_ctc lower than  810771.30
20% candidates have an expected_ctc lower than  1043658.60
25% candidates have an expected_ctc lower than  1306277.50
35% candidates have an expected_ctc lower than  1707332.90
50% candidates have an expected_ctc lower than  2252136.50
65% candidates have an expected_ctc lower than  2752874.85
75% candidates have an expected_ctc lower than  3051353.75
85% candidates have an expected_ctc lower than  3371657.20
90% candidates have an expected_ctc lower than  3796165.40
95% candidates have an expected_ctc lower than  4360145.10
```

In [62]:
```
# Calculating number of candidates in each Expected_CTC range:

plt.figure(figsize=(12,8))

ctc_range = pd.cut(expected_ctc["Expected_CTC"],
                   bins=[0, 500000, 1000000, 1500000, 2000000, 25(
                   labels=["0-500000", "500000-1000000", "1000000-
expected_ctc["ctc_range"] = ctc_range
expected_ctc["ctc_range"].value_counts().sort_index().plot(kind="ba
plt.title("Number of candidates in each CTC Range",color='blue',fon
plt.xlabel("Expected_CTC range", color="blue",size=16)
plt.xticks(rotation=45)
plt.ylabel("Count of candidates",color="blue",size=16)
plt.show()
```



In [48]:
```
# Converting year_of_passing variables to categorical type:

expected_ctc['Passing_Year_Of_Graduation']=expected_ctc['Passing_Ye
expected_ctc['Passing_Year_Of_PG']=expected_ctc['Passing_Year_Of_PG
expected_ctc['Passing_Year_Of_PHD']=expected_ctc['Passing_Year_Of_P
expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 27 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Total_Experience                    25000 non-null  int64
 1   Total_Experience_in_field_applied   25000 non-null  int64
 2   Department                          22222 non-null  object
 3   Role                                24037 non-null  object
 4   Industry                            24092 non-null  object
 5   Organization                        24092 non-null  object
 6   Designation                         21871 non-null  object
 7   Education                           25000 non-null  object
 8   Graduation_Specialization           18820 non-null  object
 9   University_Grad                     18820 non-null  object
 10  Passing_Year_Of_Graduation          18820 non-null  object
 11  PG_Specialization                   17308 non-null  object
 12  University_PG                       17308 non-null  object
 13  Passing_Year_Of_PG                  17308 non-null  object
 14  PHD_Specialization                  13119 non-null  object
 15  University_PHD                      13119 non-null  object
 16  Passing_Year_Of_PHD                 13119 non-null  object
 17  Curent_Location                     25000 non-null  object
 18  Preferred_location                  25000 non-null  object
 19  Current_CTC                         25000 non-null  int64
 20  Inhand_Offer                        25000 non-null  object
 21  Last_Appraisal_Rating               24092 non-null  object
 22  No_Of_Companies_worked              25000 non-null  int64
 23  Number_of_Publications              25000 non-null  int64
 24  Certifications                      25000 non-null  int64
 25  International_degree_any            25000 non-null  int64
 26  Expected_CTC                        25000 non-null  int64
dtypes: int64(8), object(19)
memory usage: 5.1+ MB
```

In [49]:
```


ization','University_PG','Passing_Year_Of_PG','PHD_Specialization','
t_CTC', 'Expected_CTC']
```

In [113]:
```
### Univariate analysis of continuous variables:
# Checking unique values and presence of any unwanted characters:

for column in expected_ctc[numerical]:
    print(column.upper(),': ',expected_ctc[column].nunique())
    print(expected_ctc[column].value_counts().sort_values(ascending
    print('\n')
```

```
9      954
10     867
11     776
12     711
14     624
13     610
15     539
16     498
17     381
18     329
19     317
20     245
21     200
22     146
23     109
24      82
25      37
Name: Total_Experience_in_field_applied, dtype: int64
```

In [50]:
```
# Checking missing values in continuous variables:

expected_ctc[numerical].isna().sum().sort_values(ascending=False)
```

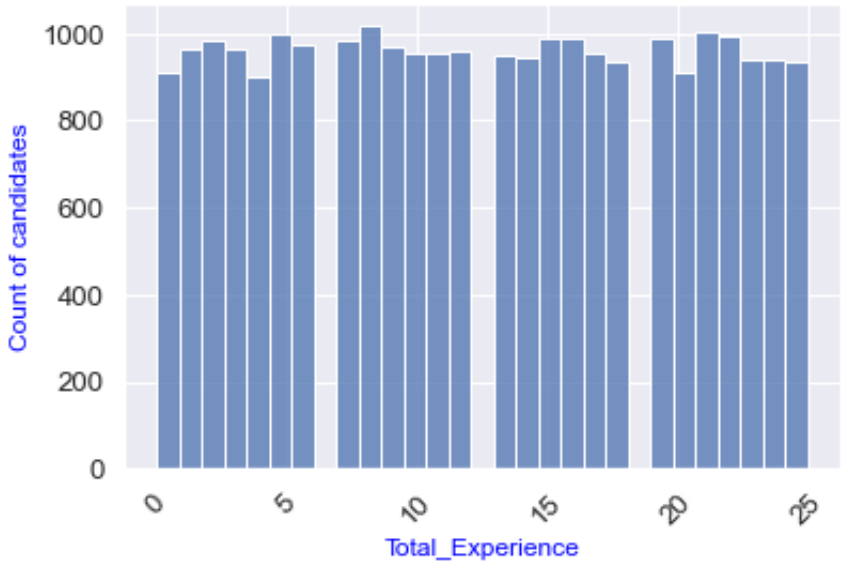
Out[50]:
```
Total_Experience                    0
Total_Experience_in_field_applied   0
No_Of_Companies_worked              0
Number_of_Publications              0
Certifications                      0
International_degree_any            0
Current_CTC                         0
Expected_CTC                        0
dtype: int64
```

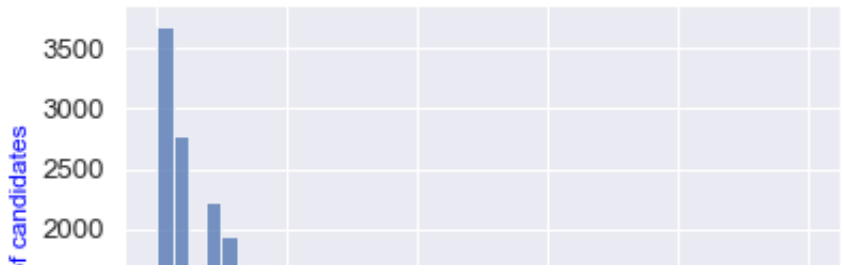
In [24]:
```
# Statistical summary of continuous variables:

expected_ctc[numerical].describe().T
```

Out[24]:

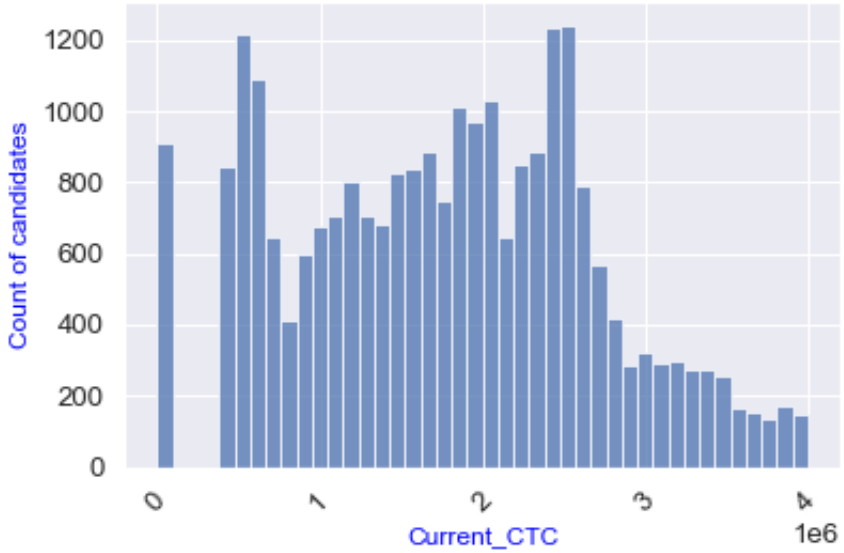
	count	mean	std	min	25%
Total_Experience	25000.0	1.249308e+01	7.471398e+00	0.0	6.0
Total_Experience_in_field_applied	25000.0	6.258200e+00	5.819513e+00	0.0	1.0
No_Of_Companies_worked	25000.0	3.482040e+00	1.690335e+00	0.0	2.0
Number_of_Publications	25000.0	4.089040e+00	2.606612e+00	0.0	2.0
Certifications	25000.0	7.736800e-01	1.199449e+00	0.0	0.0
International_degree_any	25000.0	8.172000e-02	2.739431e-01	0.0	0.0
Current_CTC	25000.0	1.760945e+06	9.202125e+05	0.0	1027311.5
Expected_CTC	25000.0	2.250155e+06	1.160480e+06	203744.0	1306277.5

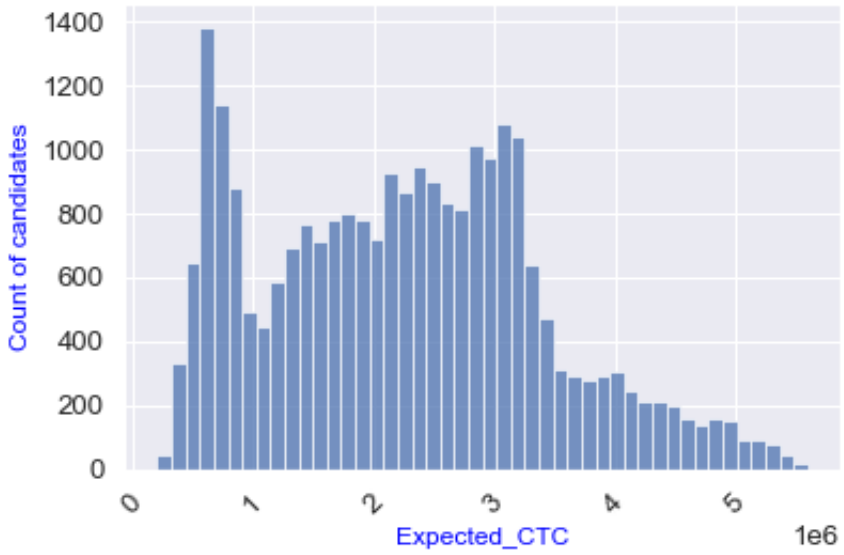
In [23]:
```
# Checking distribution of continuous variables:

for column in expected_ctc[numerical]:
    sns.histplot(expected_ctc[column])
    plt.xlabel(column,color="blue",fontsize=12)
    plt.xticks(rotation=45)
    plt.ylabel("Count of candidates",color="blue",fontsize=12)
    plt.show()
```






Count o
1500
1000
500
0
0
5
10
15
20
25
Total_Experience_in_field_applied


Count of candidates
1200
1000
800
600
400
200
0
0
1
2
3
4
Current_CTC
1e6

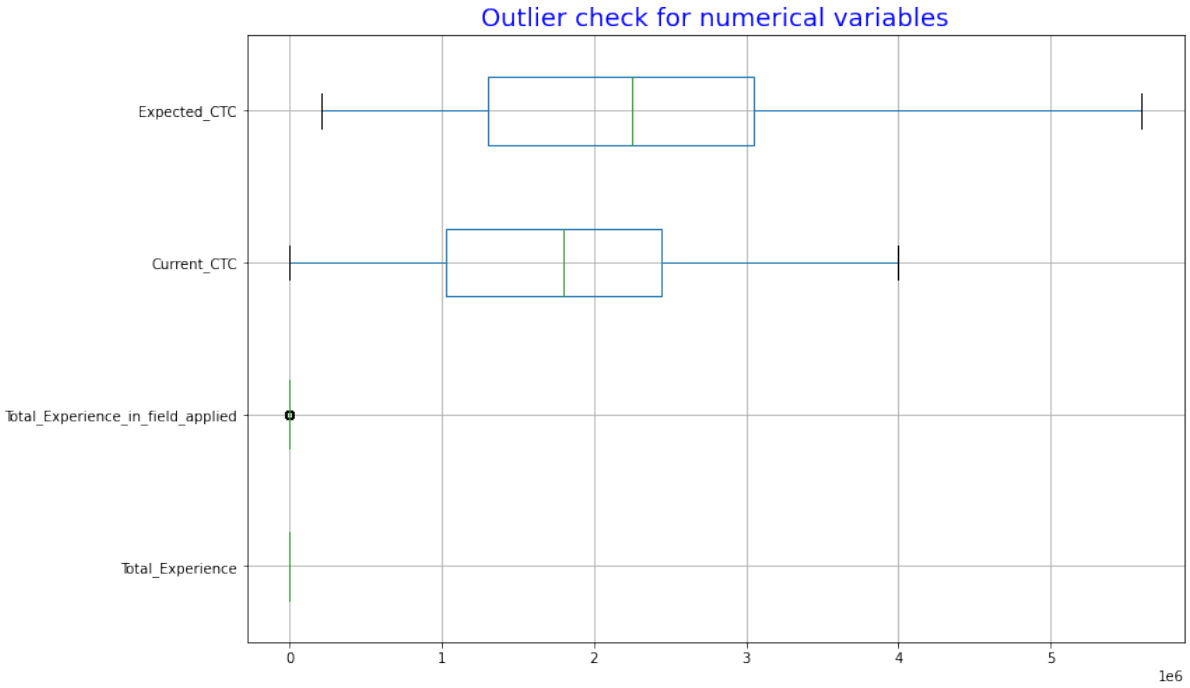

Count of candidates
1400
1200
1000
800
600
400
200
0
0
1
2
3
4
5
Expected_CTC
1e6

In [51]:
```
# From above representaton, we can say that variables 'No_Of_Compan

expected_ctc['No_Of_Companies_worked']=expected_ctc['No_Of_Companie
expected_ctc['Number_of_Publications']=expected_ctc['Number_of_Publ
expected_ctc['Certifications']=expected_ctc['Certifications'].astyp
expected_ctc['International_degree_any']=expected_ctc['Internationa
expected_ctc.info()
```

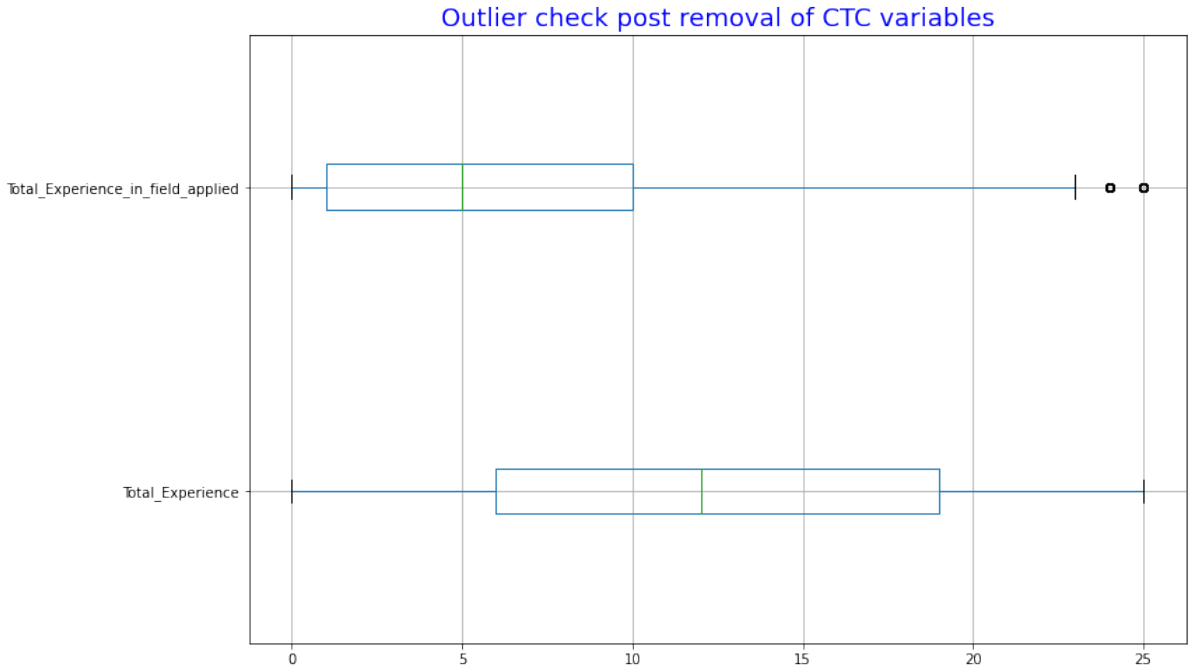
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 27 columns):
 #   Column                              Non-Null Count  Dtype 
---  ------                              --------------  ----- 
 0   Total_Experience                    25000 non-null  int64 
 1   Total_Experience_in_field_applied   25000 non-null  int64 
 2   Department                          22222 non-null  object
 3   Role                                24037 non-null  object
 4   Industry                            24092 non-null  object
 5   Organization                        24092 non-null  object
 6   Designation                         21871 non-null  object
 7   Education                           25000 non-null  object
 8   Graduation_Specialization           18820 non-null  object
 9   University_Grad                     18820 non-null  object
 10  Passing_Year_Of_Graduation          18820 non-null  object
 11  PG_Specialization                   17308 non-null  object
 12  University_PG                       17308 non-null  object
 13  Passing_Year_Of_PG                  17308 non-null  object
 14  PHD_Specialization                  13119 non-null  object
 15  University_PHD                      13119 non-null  object
 16  Passing_Year_Of_PHD                 13119 non-null  object
 17  Curent_Location                     25000 non-null  object
 18  Preferred_location                  25000 non-null  object
 19  Current_CTC                         25000 non-null  int64 
 20  Inhand_Offer                        25000 non-null  object
 21  Last_Appraisal_Rating               24092 non-null  object
 22  No_Of_Companies_worked              25000 non-null  object
 23  Number_of_Publications              25000 non-null  object
 24  Certifications                      25000 non-null  object
 25  International_degree_any            25000 non-null  object
 26  Expected_CTC                        25000 non-null  int64 
dtypes: int64(4), object(23)
memory usage: 5.1+ MB
```

In [52]:
```
# Redefining variable set:

categorical=['Department','Role','Industry','Organization','Designa
numerical=['Total_Experience','Total_Experience_in_field_applied',
```

In [28]:
```
# Checking for outliers using boxplots:

plt.figure(figsize=(12,8))
expected_ctc[numerical].boxplot(vert=0)
plt.title("Outlier check for numerical variables",color='blue',font
plt.show()
```



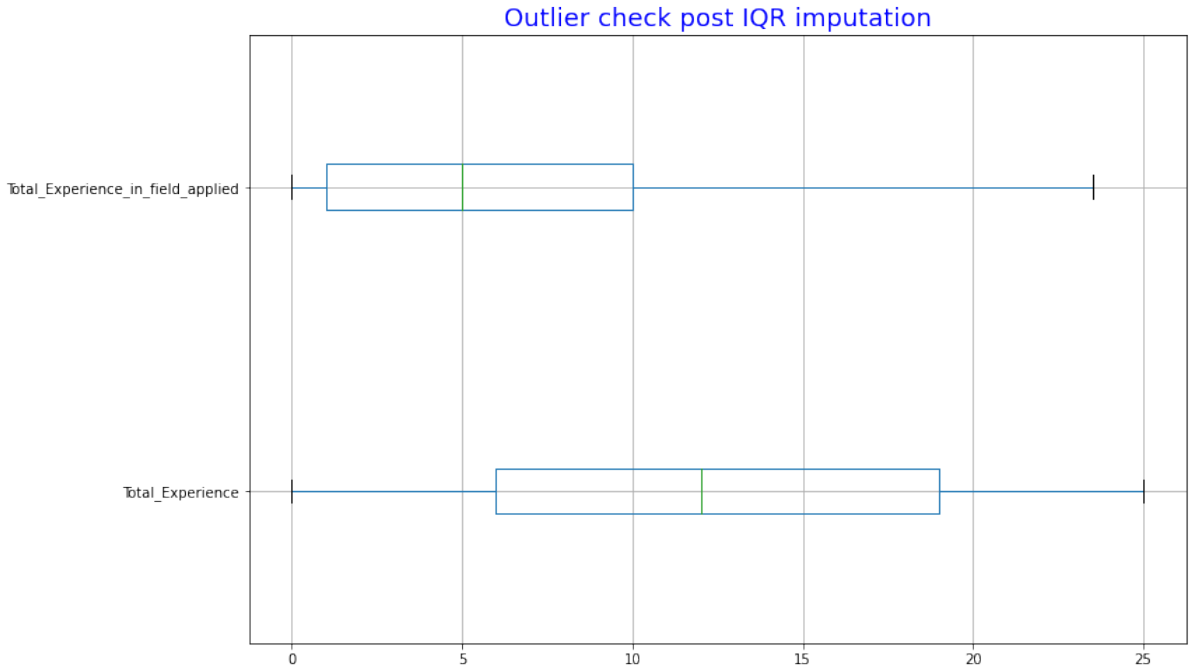
In [29]:
```
# Boxplots without 'Expected_CTC' & 'Current_CTC' to get a clearer

plt.figure(figsize=(12,8))
expected_ctc[numerical].drop(['Expected_CTC','Current_CTC'],axis=1)
plt.title('Outlier check post removal of CTC variables',color='blue
plt.show()
```



In [53]:
```
# Since 'Total_Experience_in_field_applied' has few outliers, we wi

def remove_outlier(col):
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range= Q1-(1.5 * IQR)
    upper_range= Q3+(1.5 * IQR)
    return lower_range, upper_range
```

In [54]:
```
ce_in_field_applied'])
=np.where(expected_ctc['Total_Experience_in_field_applied']>ur,ur,ex
=np.where(expected_ctc['Total_Experience_in_field_applied']<lr,lr,ex
```

In [32]:
```
#Re-checking for outliers post imputation:

plt.figure(figsize=(12,8))
expected_ctc[numerical].drop(['Expected_CTC','Current_CTC'],axis=1)
plt.title('Outlier check post IQR imputation',color='blue',fontsize
plt.show()
```



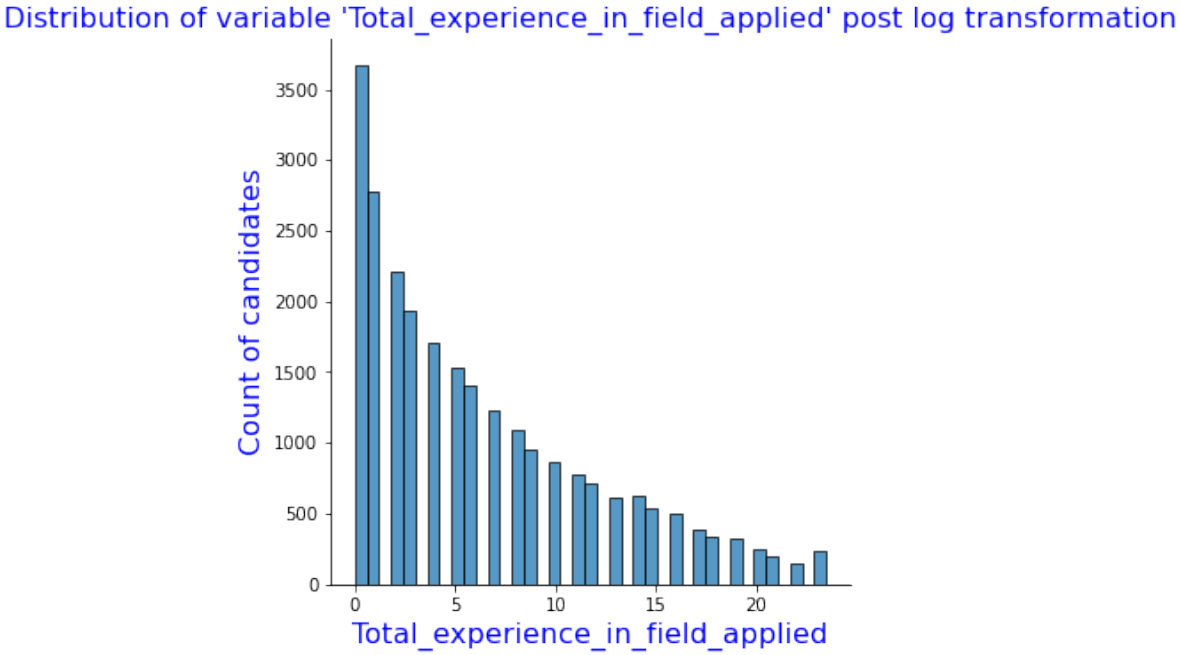
In [33]:
```
# Checking for skewness in the continuous variables:

expected_ctc[numerical].skew()
```

Out[33]:
```
Total_Experience                    0.004109
Total_Experience_in_field_applied   0.951124
Current_CTC                         0.097643
Expected_CTC                        0.331972
dtype: float64
```

In [34]:
```
# Very high skewness in 'Total_Experience_in_field_applied' - 95%.

log_tot_exp_in_field=np.log(expected_ctc['Total_Experience_in_field
sns.displot(expected_ctc['Total_Experience_in_field_applied'])
plt.title("Distribution of variable 'Total_experience_in_field_appl
plt.xlabel("Total_experience_in_field_applied", color="blue",size=1
plt.ylabel("Count of candidates",color="blue",size=16)
```

```
/Users/sabita/opt/anaconda3/lib/python3.8/site-packages/pandas/cor
e/arraylike.py:397: RuntimeWarning: divide by zero encountered in
log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[34]:
```
Text(-2.575000000000003, 0.5, 'Count of candidates')
```



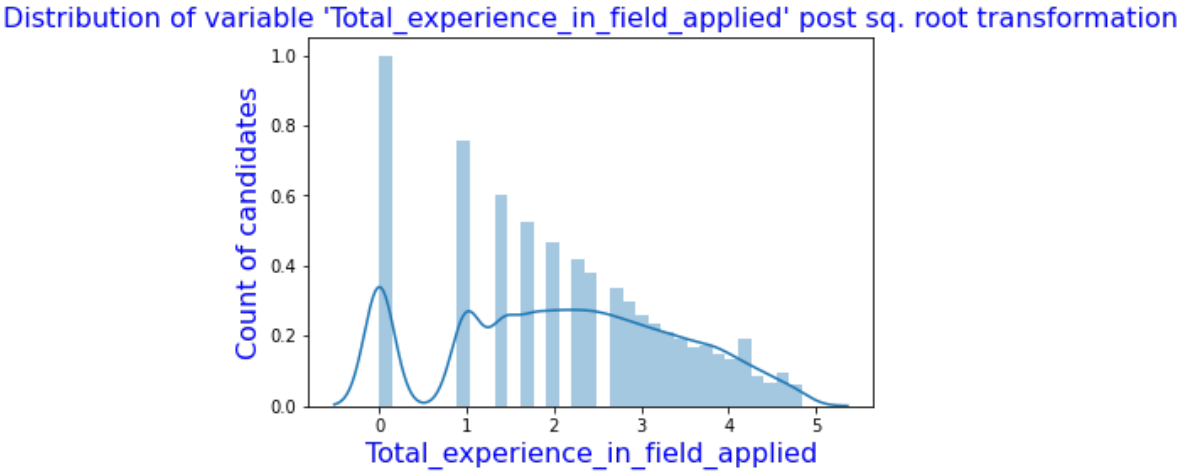
In [35]:
```
#Log tranformation has not helped the skewness - let us try boxcox

from scipy.stats import boxcox
expected_ctc['Total_Experience_in_field_applied'],lmbda=boxcox(expe
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
<ipython-input-35-cb753e7db60e> in <module>
      2
      3 from scipy.stats import boxcox
----> 4 expected_ctc['Total_Experience_in_field_applied'],lmbda=bo
xcox(expected_ctc['Total_Experience_in_field_applied'],lmbda=None)

~/opt/anaconda3/lib/python3.8/site-packages/scipy/stats/morestats.
py in boxcox(x, lmbda, alpha)
   1041
   1042     if any(x <= 0):
-> 1043         raise ValueError("Data must be positive.")
   1044
   1045     if lmbda is not None:  # single transformation

ValueError: Data must be positive.
```

In [36]:
```
# Since boxcox does not work in this case, let us try square root t

sqrt=expected_ctc['Total_Experience_in_field_applied']**(.5)
sns.distplot(sqrt)
plt.title("Distribution of variable 'Total_experience_in_field_appl
plt.xlabel("Total_experience_in_field_applied", color="blue",size=1
plt.ylabel("Count of candidates",color="blue",size=16)
```

```
/Users/sabita/opt/anaconda3/lib/python3.8/site-packages/seaborn/di
stributions.py:2619: FutureWarning: `distplot` is a deprecated fun
ction and will be removed in a future version. Please adapt your c
ode to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms)
.
  warnings.warn(msg, FutureWarning)
```

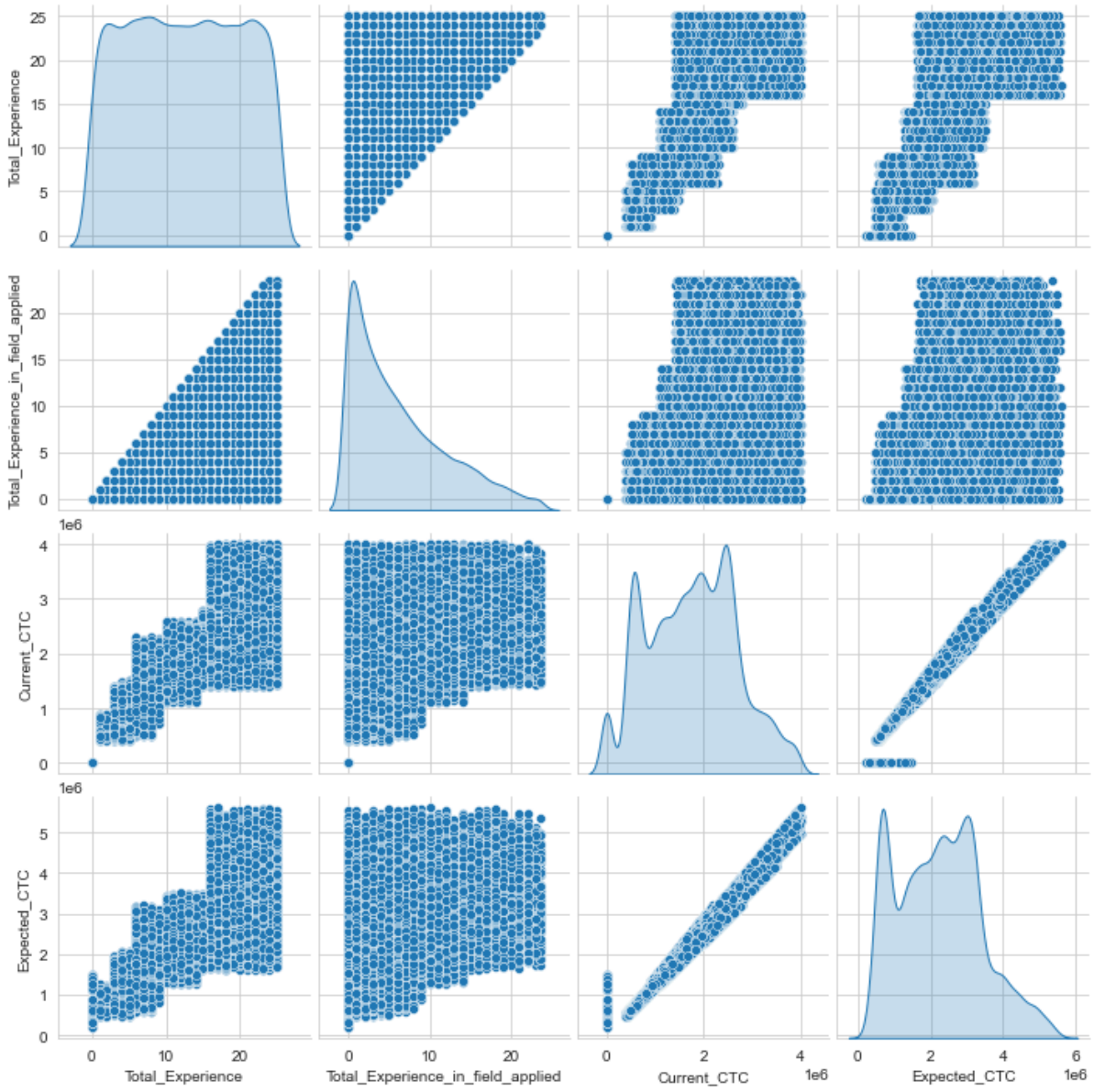
Out[36]:
```
Text(0, 0.5, 'Count of candidates')
```



In [ ]:
```
# Since transformation has not improved the distribution of 'Total_
```

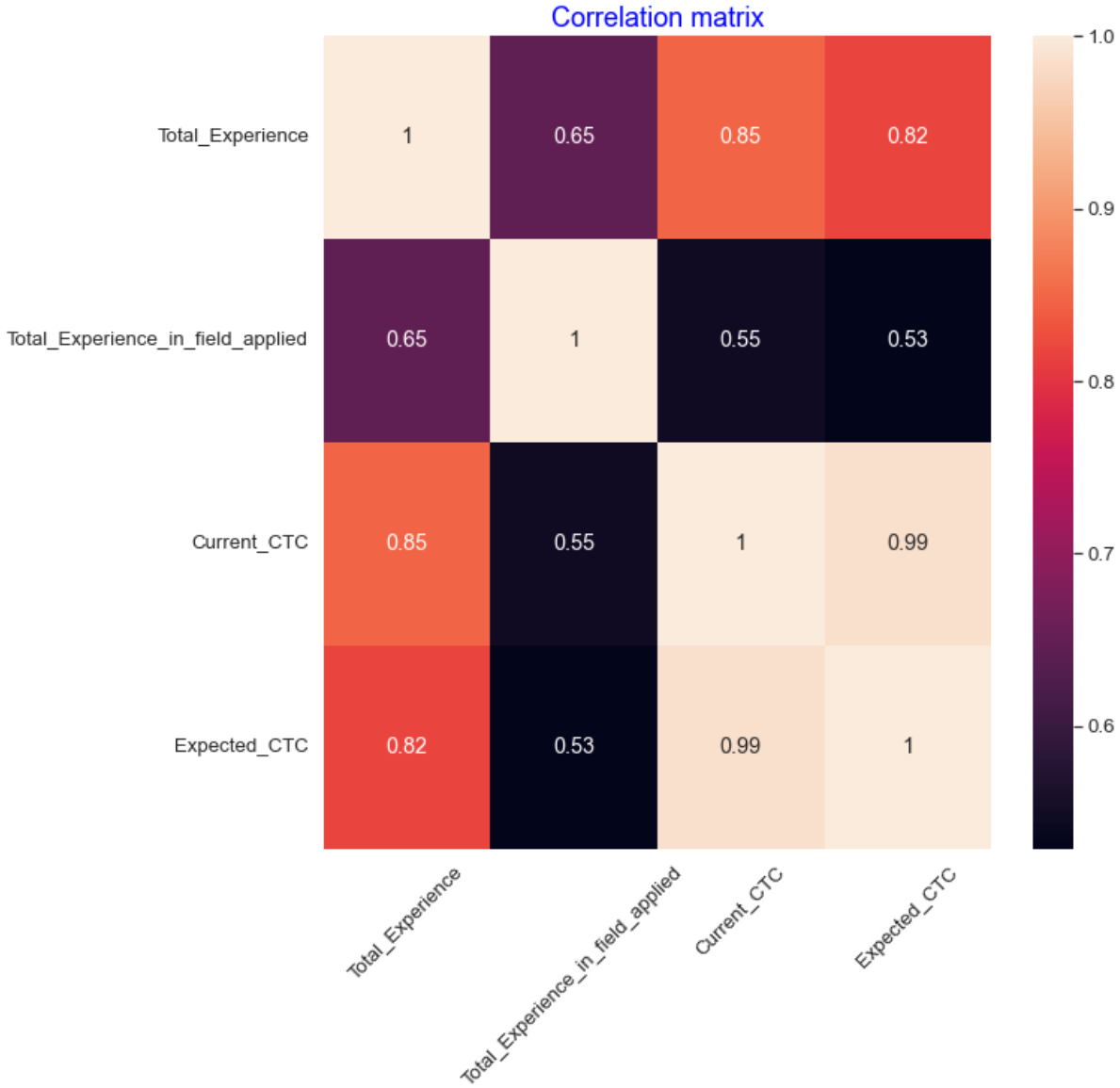
In [ ]:
```
### Bivariate analysis of continuous variables:
```

In [37]:
```
# Checking relation between continuous variables:

sns.set_style('whitegrid')
sns.pairplot(data=expected_ctc[numerical],diag_kind="kde")
plt.show()
```



In [22]:
```
# Using a heatmap to visualize correlation:

plt.figure(figsize=(10,10))
sns.set(font_scale=1.2)
sns.heatmap(expected_ctc[numerical].corr(),annot=True)
plt.title('Correlation matrix',color='blue',fontsize=18)
plt.xticks(rotation=45)
```

Out[22]:
```
(array([0.5, 1.5, 2.5, 3.5]),
 [Text(0.5, 0, 'Total_Experience'),
  Text(1.5, 0, 'Total_Experience_in_field_applied'),
  Text(2.5, 0, 'Current_CTC'),
  Text(3.5, 0, 'Expected_CTC')])
```



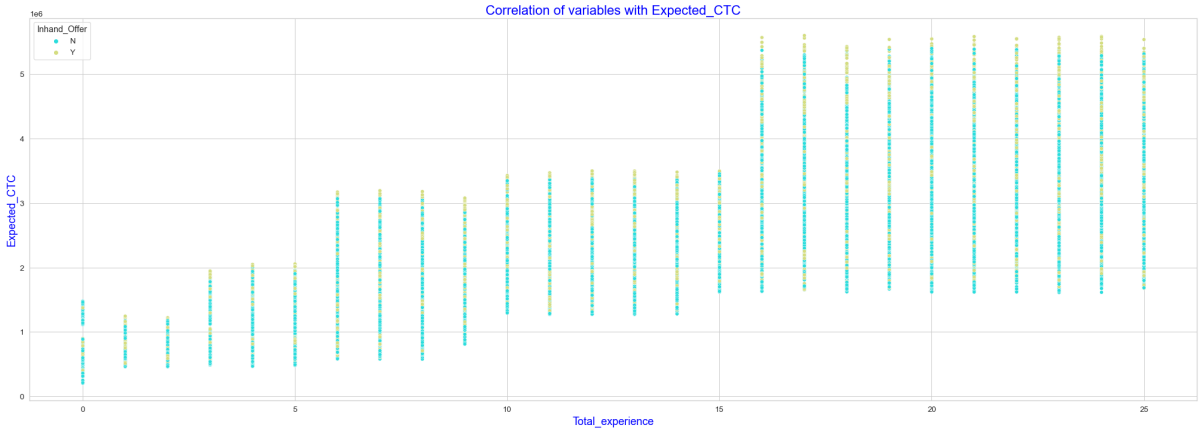
In [62]:
```
expected_ctc[numerical].corr()
```

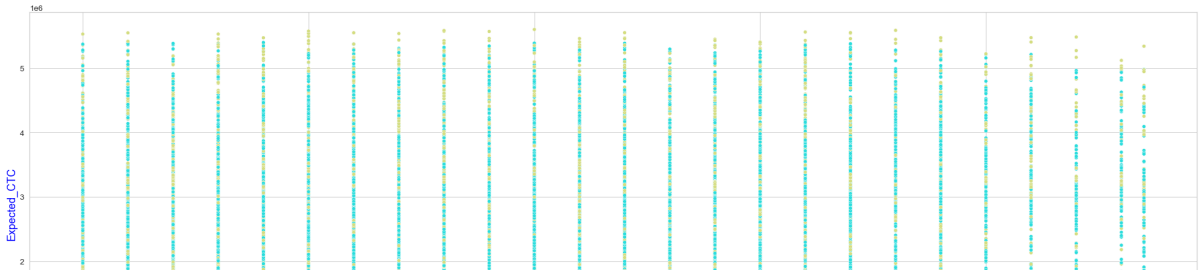
Out[62]:

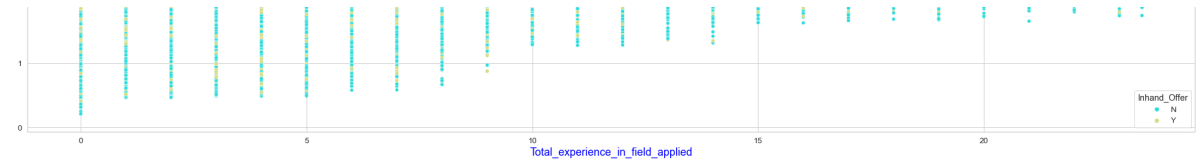
	Total_Experience	Total_Experience_in_field_applied	Current
Total_Experience	1.000000	0.645135	0.8
Total_Experience_in_field_applied	0.645135	1.000000	0.5
Current_CTC	0.846476	0.548017	1.0
Expected_CTC	0.816593	0.529115	0.9

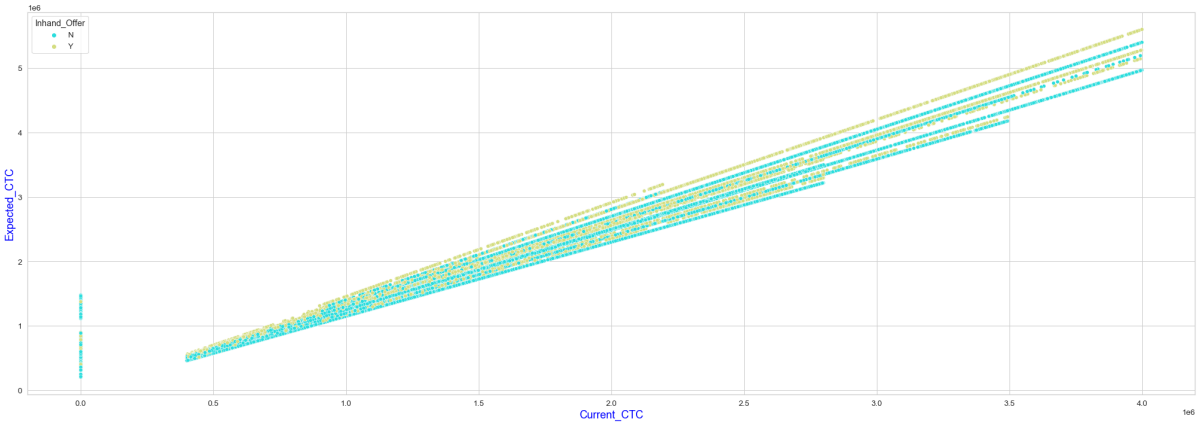
In [ ]:
```
# Highest correlation to target variable - 'Total_Experience' (~82%
```

In [39]:
```
# Scatterplots against 'Expected_CTC':

sns.set_style('whitegrid')
plt.figure(figsize=(35,40))
plt.subplot(3,1,1)
sns.scatterplot(x=expected_ctc['Total_Experience'],y=expected_ctc['
plt.title("Correlation of variables with Expected_CTC",color="blue"
plt.xlabel("Total_experience", color="blue",size=18)
plt.ylabel("Expected_CTC",color="blue",size=18)
plt.subplot(3,1,2)
sns.scatterplot(x=expected_ctc['Total_Experience_in_field_applied']
plt.xlabel("Total_experience_in_field_applied", color="blue",size=1
plt.ylabel("Expected_CTC",color="blue",size=18)
plt.subplot(3,1,3)
sns.scatterplot(x=expected_ctc['Current_CTC'],y=expected_ctc['Expec
plt.xlabel("Current_CTC", color="blue",size=18)
plt.ylabel("Expected_CTC",color="blue",size=18)
plt.show()
```









In [ ]:
```
### Univariate analysis of discrete variables:
```

In [55]:
```
# Checking for unique categorical values: (to check data hygiene)

for column in expected_ctc[categorical]:
    print(column.upper(),': ',expected_ctc[column].nunique())
    print(expected_ctc[column].value_counts().sort_values(ascending
    print('\n')
```

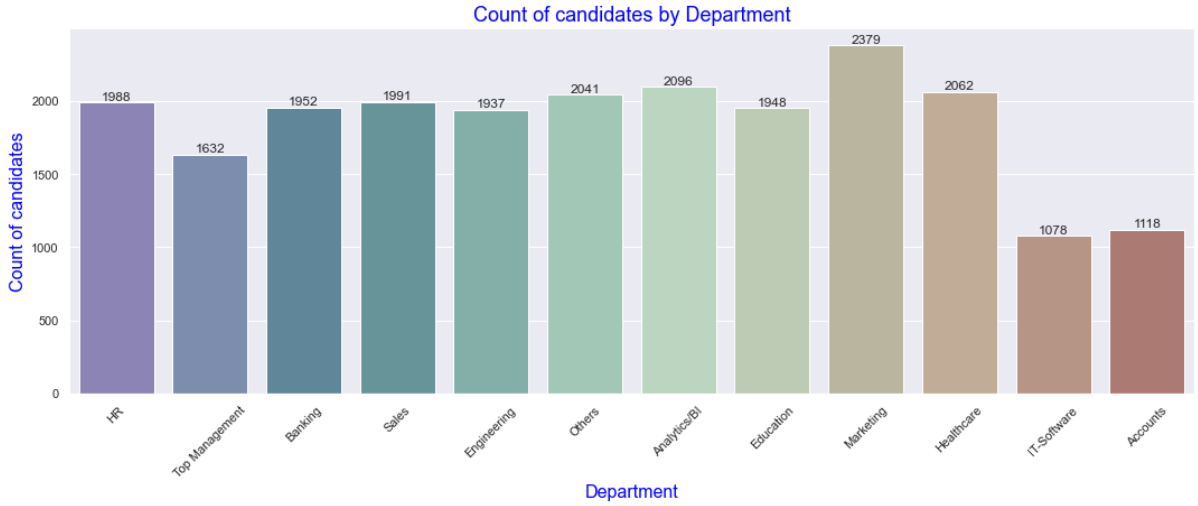
```
Name: Number_of_Publications, dtype: int64


CERTIFICATIONS :  6
0    15215
1     4644
2     2198
3     1818
4      777
5      348
Name: Certifications, dtype: int64


INTERNATIONAL_DEGREE_ANY :  2
0    22957
1     2043
Name: International_degree_any, dtype: int64
```

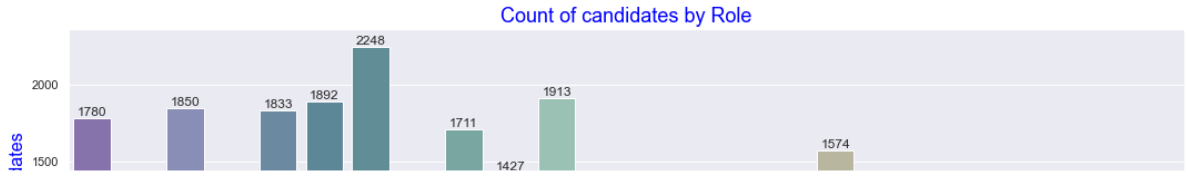
In [77]:
```
# Statistical summary of continuous variables:

expected_ctc[categorical].describe(include="all").T
```

Out[77]:

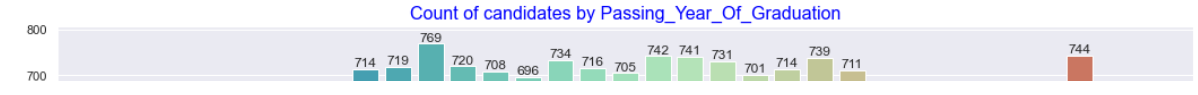
	count	unique	top	freq
Department	22222	12	Marketing	2379
Role	24037	24	Others	2248
Industry	24092	11	Training	2237
Organization	24092	16	M	1574
Designation	21871	18	HR	1648
Education	25000	4	PG	6326
Graduation_Specialization	18820	11	Chemistry	1785
University_Grad	18820	13	Bhubaneswar	1510
Passing_Year_Of_Graduation	18820.0	35.0	1997.0	769.0
PG_Specialization	17308	11	Mathematics	1800
University_PG	17308	13	Bhubaneswar	1377
Passing_Year_Of_PG	17308.0	36.0	2011.0	816.0
PHD_Specialization	13119	11	Others	1545
University_PHD	13119	13	Kolkata	1069
Passing_Year_Of_PHD	13119.0	26.0	1998.0	536.0
Curent_Location	25000	15	Bangalore	1742
Preferred_location	25000	15	Kanpur	1720
Inhand_Offer	25000	2	N	17418
Last_Appraisal_Rating	24092	5	B	5501

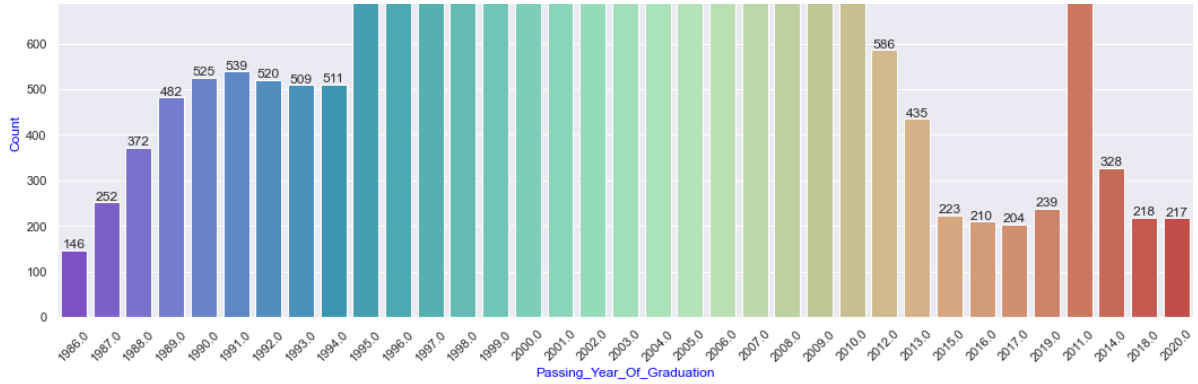
In [15]:
```
# Checking distibution of variables:

cat=["Department","Role","Education","Organization","Designation","
sns.set()

for a in expected_ctc[cat]:
    plt.figure(figsize=(18,6))
    ax=sns.countplot(x=expected_ctc[a],palette="rainbow",saturation
    plt.title("Count of candidates by "+a,color="blue",fontsize=18)
    plt.xlabel(a,color="blue",fontsize=16)
    plt.xticks(rotation=45)
    plt.ylabel("Count of candidates",color="blue",fontsize=16)
    ax.bar_label(ax.containers[0])
    plt.show()
```

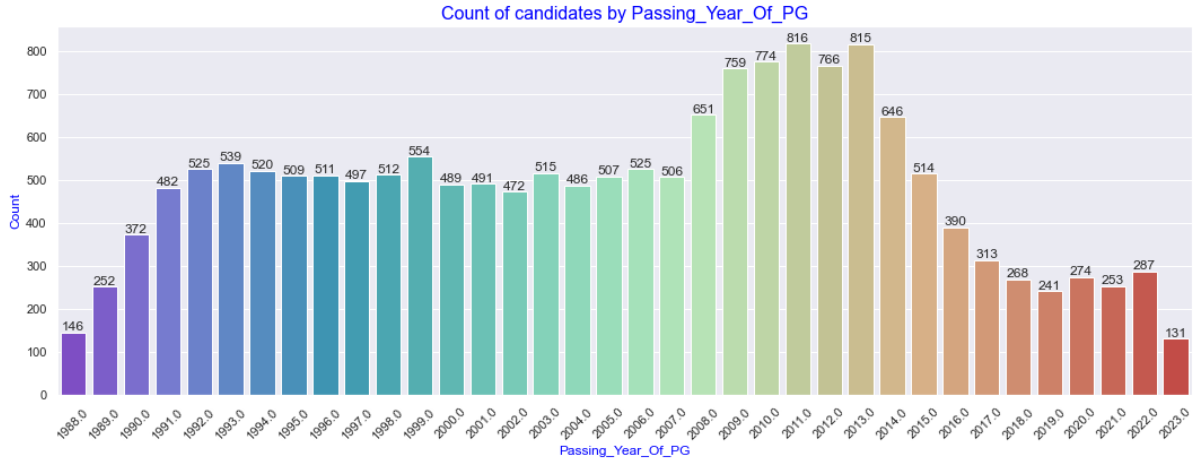


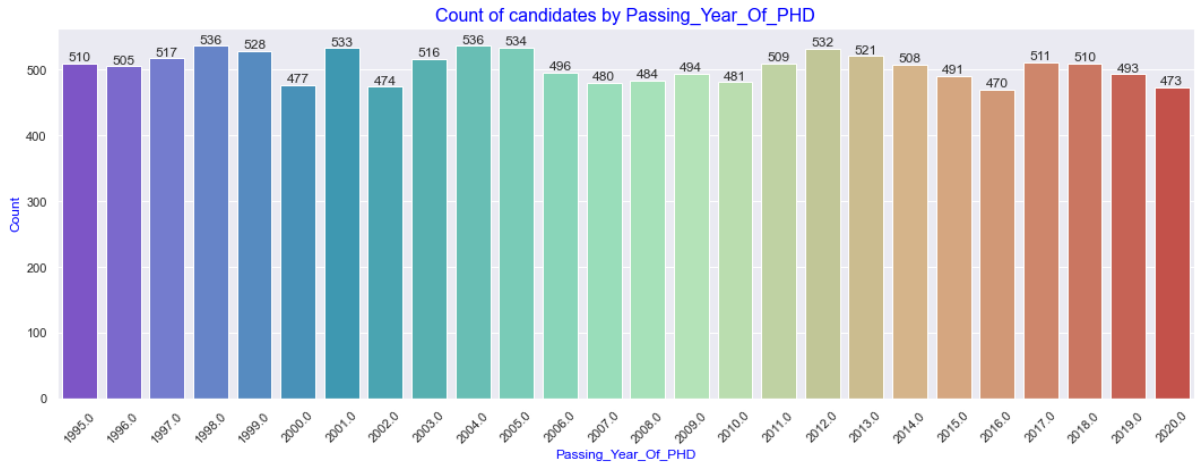


In [16]:
```
# Checking distibution of variables:

cat=["Passing_Year_Of_Graduation","Passing_Year_Of_PG","Passing_Yea

for i in expected_ctc[cat]:
    plt.figure(figsize=(18,6))
    ax=sns.countplot(x=expected_ctc[i],palette="rainbow",saturation
    plt.title("Count of candidates by "+i,color="blue",fontsize=16)
    plt.xlabel(i,color="blue",fontsize=12)
    plt.xticks(rotation=45)
    plt.ylabel("Count",color="blue",fontsize=12)
    ax.bar_label(ax.containers[0])
    plt.show()
```

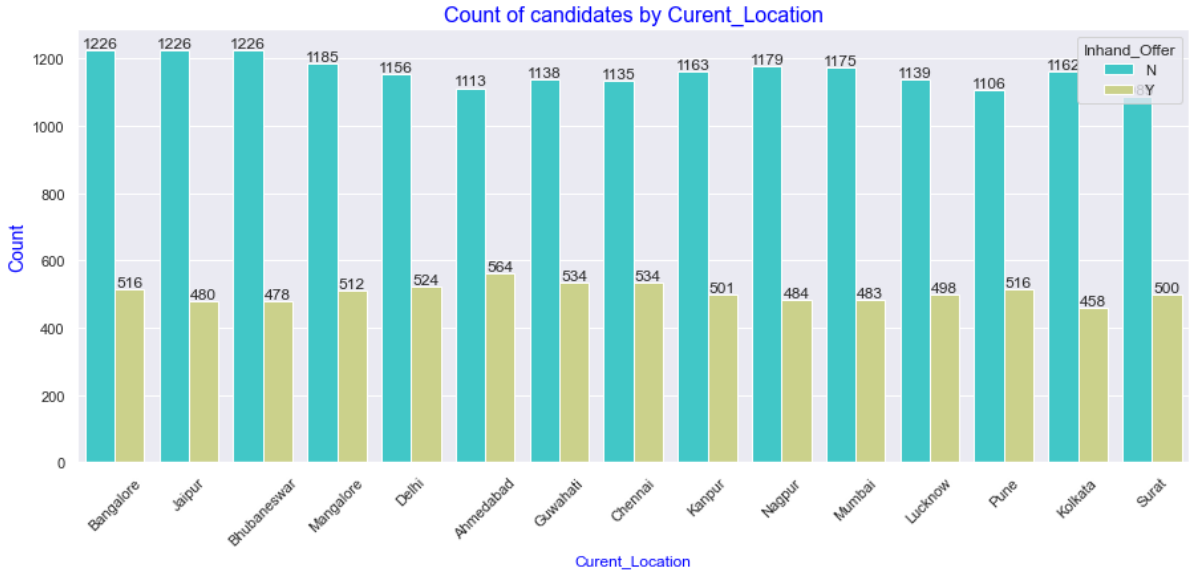


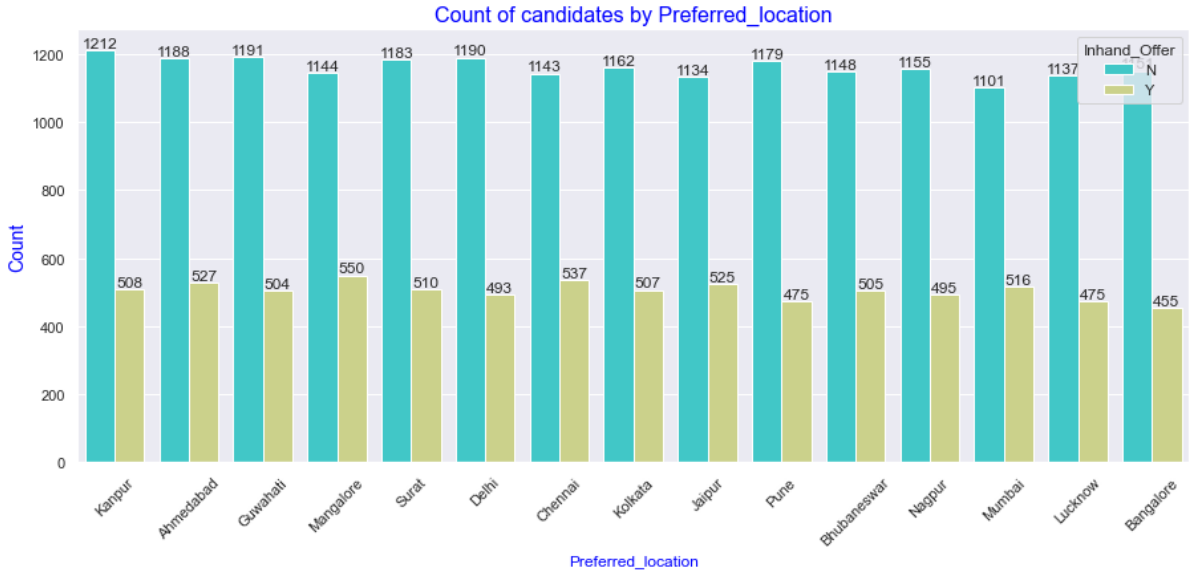

Count
Passing_Year_Of_Graduation
146 252 372 482 525 539 520 509 511 586 435 223 210 204 239 328 218 217


Count of candidates by Passing_Year_Of_PG
Count
Passing_Year_Of_PG
146 252 372 482 525 539 520 509 511 497 512 554 489 491 472 515 486 507 525 506 651 759 774 816 766 815 646 514 390 313 268 241 274 253 287 131


Count of candidates by Passing_Year_Of_PHD
Count
Passing_Year_Of_PHD
510 505 517 536 528 477 533 474 516 536 534 496 480 484 494 481 509 532 521 508 491 470 511 510 493 473

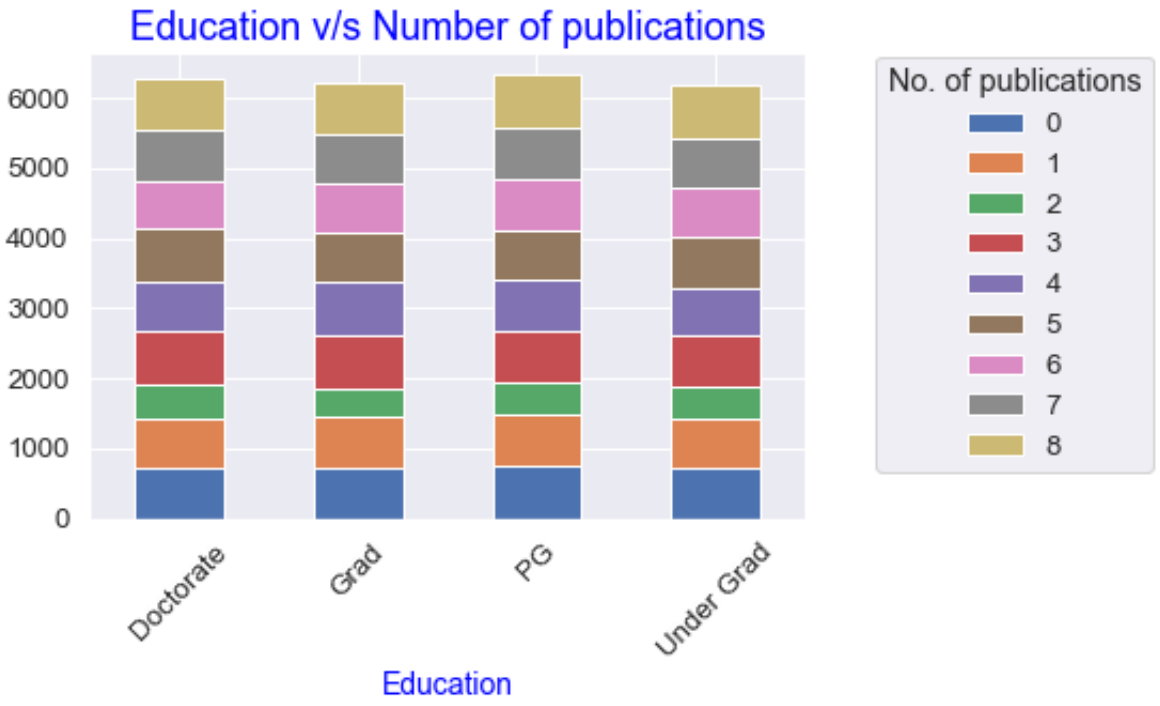
In [17]:
```
cat1=["Curent_Location","Preferred_location"]

for i in expected_ctc[cat1]:
    plt.figure(figsize=(15,6))
    ax=sns.countplot(x=expected_ctc[i],order=expected_ctc[i].value_
    plt.title("Count of candidates by "+i,color="blue",fontsize=16)
    plt.xlabel(i,color="blue",fontsize=12)
    plt.xticks(rotation=45)
    plt.ylabel("Count",color="blue",fontsize=14)
    for container in ax.containers:
        ax.bar_label(container)
    plt.show()
```





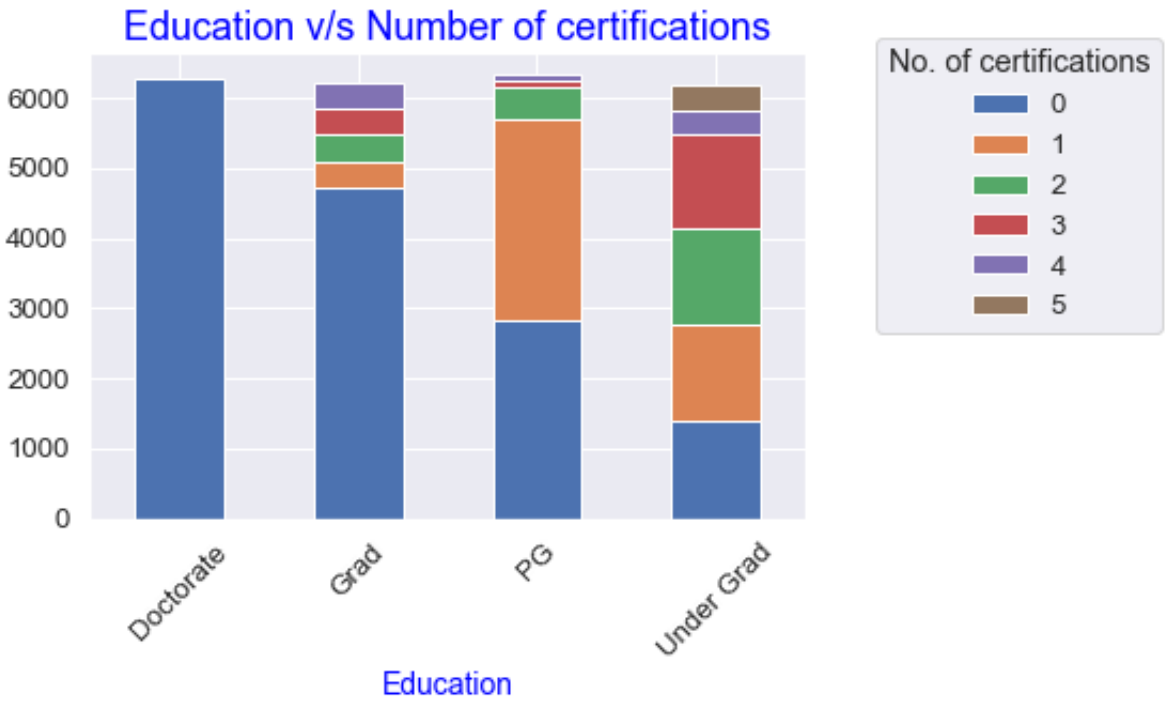
In [37]:
```
import pylab as pl

plot=pd.crosstab(expected_ctc['Education'],expected_ctc['Number_of_
plot.set_title("Education v/s Number of publications",color="blue",
plot.legend(loc=(1.1,0.1),title='No. of publications')
plt.xlabel('Education',color="blue",fontsize=14)
plt.xticks(rotation=45)
```

Out[37]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Doctorate'),
  Text(1, 0, 'Grad'),
  Text(2, 0, 'PG'),
  Text(3, 0, 'Under Grad')])
```

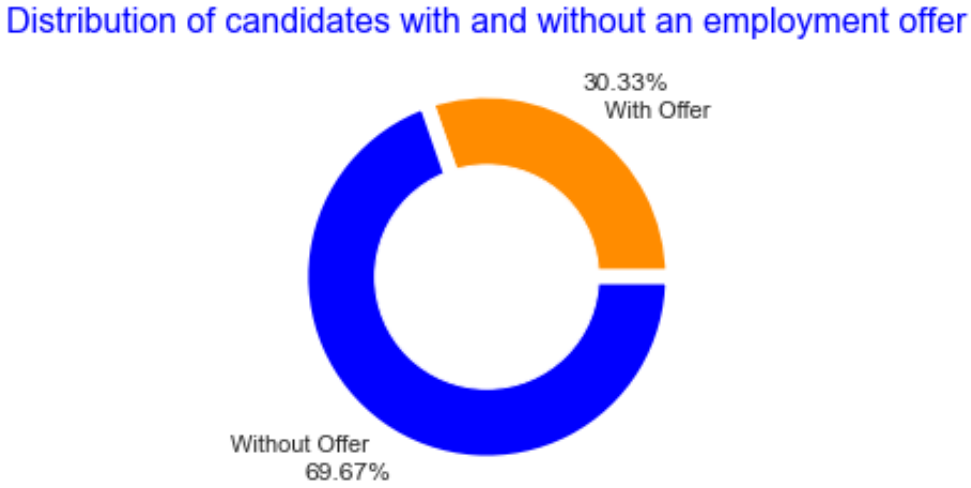


In [38]:
```
import pylab as pl
plot=pd.crosstab(expected_ctc['Education'],expected_ctc['Certificat
plot.set_title("Education v/s Number of certifications",color="blue'
plt.xlabel('Education',color="blue",fontsize=14)
plot.legend(loc=(1.1,0.4),title='No. of certifications')
plt.xticks(rotation=45)
```

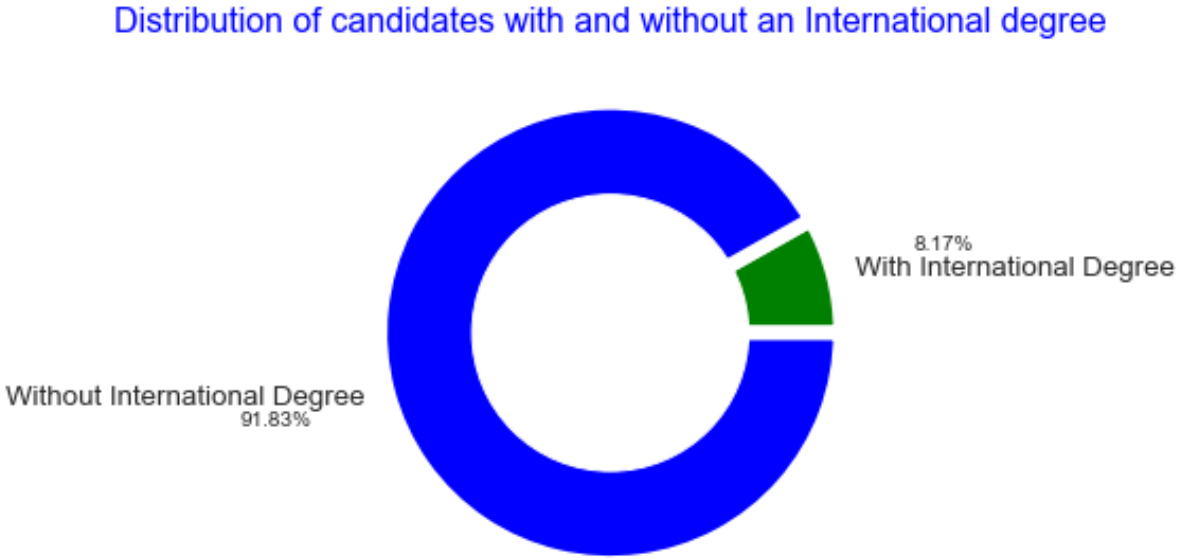
Out[38]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Doctorate'),
  Text(1, 0, 'Grad'),
  Text(2, 0, 'PG'),
  Text(3, 0, 'Under Grad')])
```



In [48]:
```
offer = expected_ctc.Inhand_Offer.value_counts()['Y']
no_offer = expected_ctc.Inhand_Offer.value_counts()['N']
names = ['With Offer','Without Offer']
size  = [offer, no_offer]

plt.pie(size, labels=names, colors=['darkorange','blue'],
        autopct='%.2f%%', pctdistance=1.28,
        wedgeprops={'linewidth':7, 'edgecolor':'white'})

# creating circle for the center of the plot to make the pie look l
my_circle = plt.Circle((0,0), 0.6, color='white')

# plot the donut chart
fig = plt.gcf()
fig.set_size_inches(4,4)
fig.gca().add_artist(my_circle)
plt.title("Distribution of candidates with and without an employmen
plt.show()
```



In [63]:
```
int_degree = expected_ctc.International_degree_any.value_counts()[1
no_int_degree = expected_ctc.International_degree_any.value_counts(
names = ['With International Degree','Without International Degree'
size  = [int_degree, no_int_degree ]

plt.pie(size, labels=names, colors=['green','blue'],
        autopct='%.2f%%', pctdistance=1.5,
        wedgeprops={'linewidth':7, 'edgecolor':'white'})

# creating circle for the center of the plot to make the pie look l
my_circle = plt.Circle((0,0), 0.6, color='white')

# plot the donut chart
fig = plt.gcf()
fig.set_size_inches(5,5)
fig.gca().add_artist(my_circle)
plt.title("Distribution of candidates with and without an Internati
plt.show()
```



In [56]:
```
# Checking for missing/null values in categorical columns:

expected_ctc[categorical].isna().sum().sort_values(ascending=False)
```

Out[56]:
```
Passing_Year_Of_PHD           11881
University_PHD                11881
PHD_Specialization            11881
Passing_Year_Of_PG             7692
University_PG                  7692
PG_Specialization              7692
Graduation_Specialization      6180
University_Grad                6180
Passing_Year_Of_Graduation     6180
Designation                    3129
Department                     2778
Role                            963
Organization                    908
Industry                        908
Last_Appraisal_Rating           908
Education                         0
Curent_Location                   0
Preferred_location                0
Inhand_Offer                      0
No_Of_Companies_worked            0
Number_of_Publications            0
Certifications                    0
International_degree_any          0
dtype: int64
```

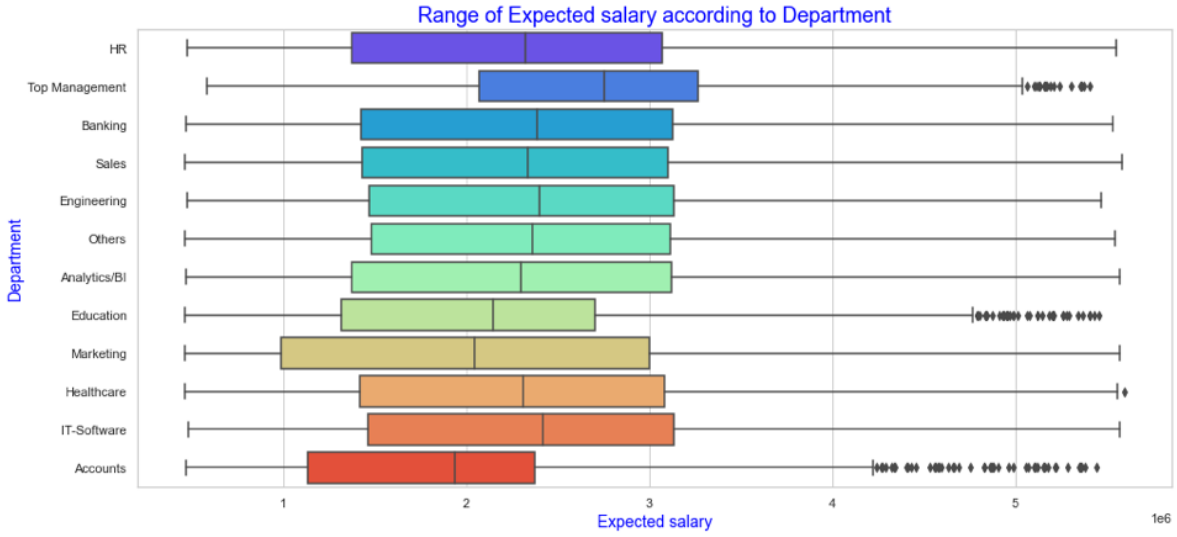
In [ ]:
```
# High number of missing values in several variables- further analy
```

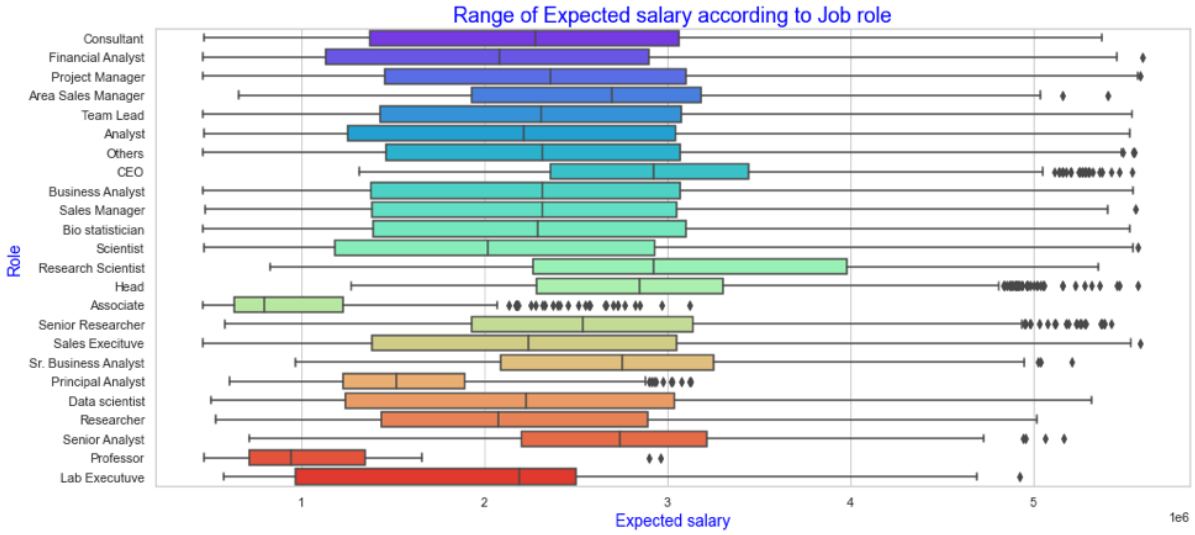
In [ ]:
```
### Bivariate analysis of categorical variables:
```

In [ ]:
```
# Checking relation of variables with target variable:
```

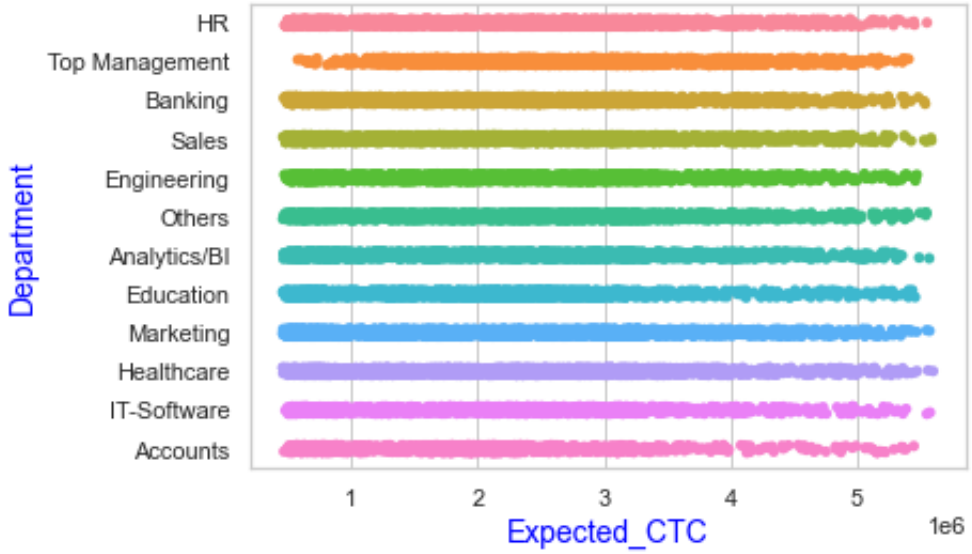
In [80]:
```
s.set_style('whitegrid')
t.figure(figsize=(16,16))
t.subplot(2,1,1)
s.boxplot(x=expected_ctc['Expected_CTC'],y=expected_ctc['Department'
t.title("Range of Expected salary according to Department",color="bl
t.ylabel("Department",color="blue",fontsize=14)
t.xlabel("Expected salary",color="blue",fontsize=14)
t.subplot(2,1,2)
s.boxplot(x=expected_ctc['Expected_CTC'],y=expected_ctc['Role'],data
t.title("Range of Expected salary according to Job role",color="blue
t.ylabel("Role",color="blue",fontsize=14)
t.xlabel("Expected salary",color="blue",fontsize=14)
```

Out[80]: Text(0.5, 0, 'Expected salary')

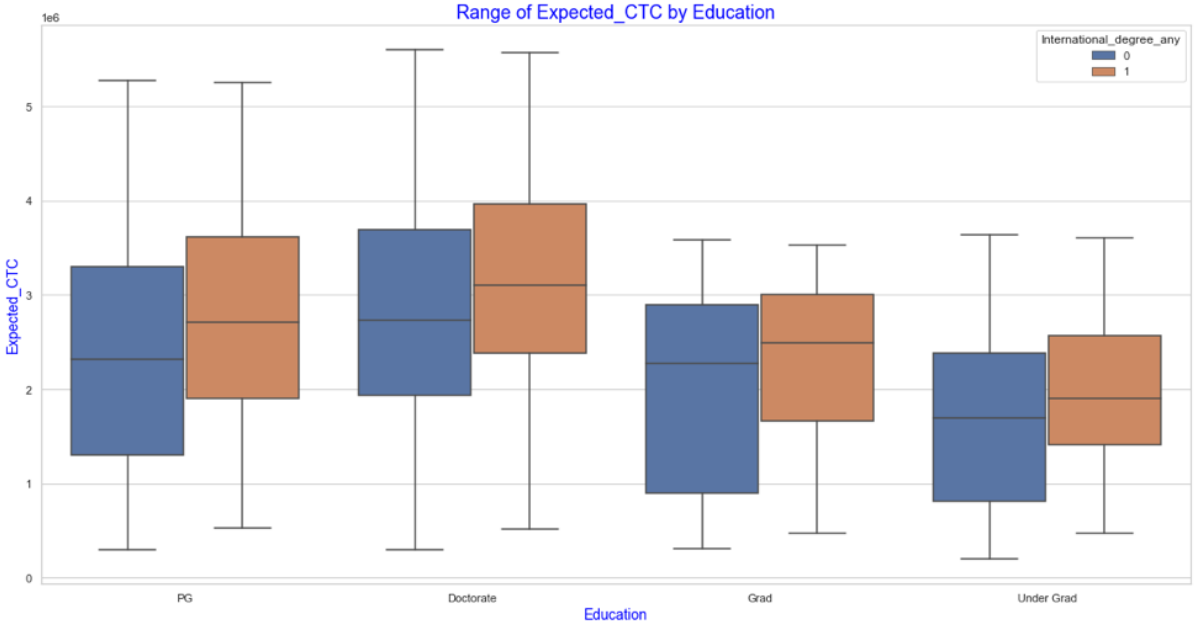




In [81]:
```
# Creating subplots for all categorical variables with target varia

cat=['Department','Role','Industry','Organization','Designation','E

for i in cat:
    sns.stripplot(x='Expected_CTC',y=i,data=expected_ctc,orient='h'
    plt.ylabel(i,color="blue",fontsize=14)
    plt.xlabel("Expected_CTC",color="blue",fontsize=14)
    plt.show()
```



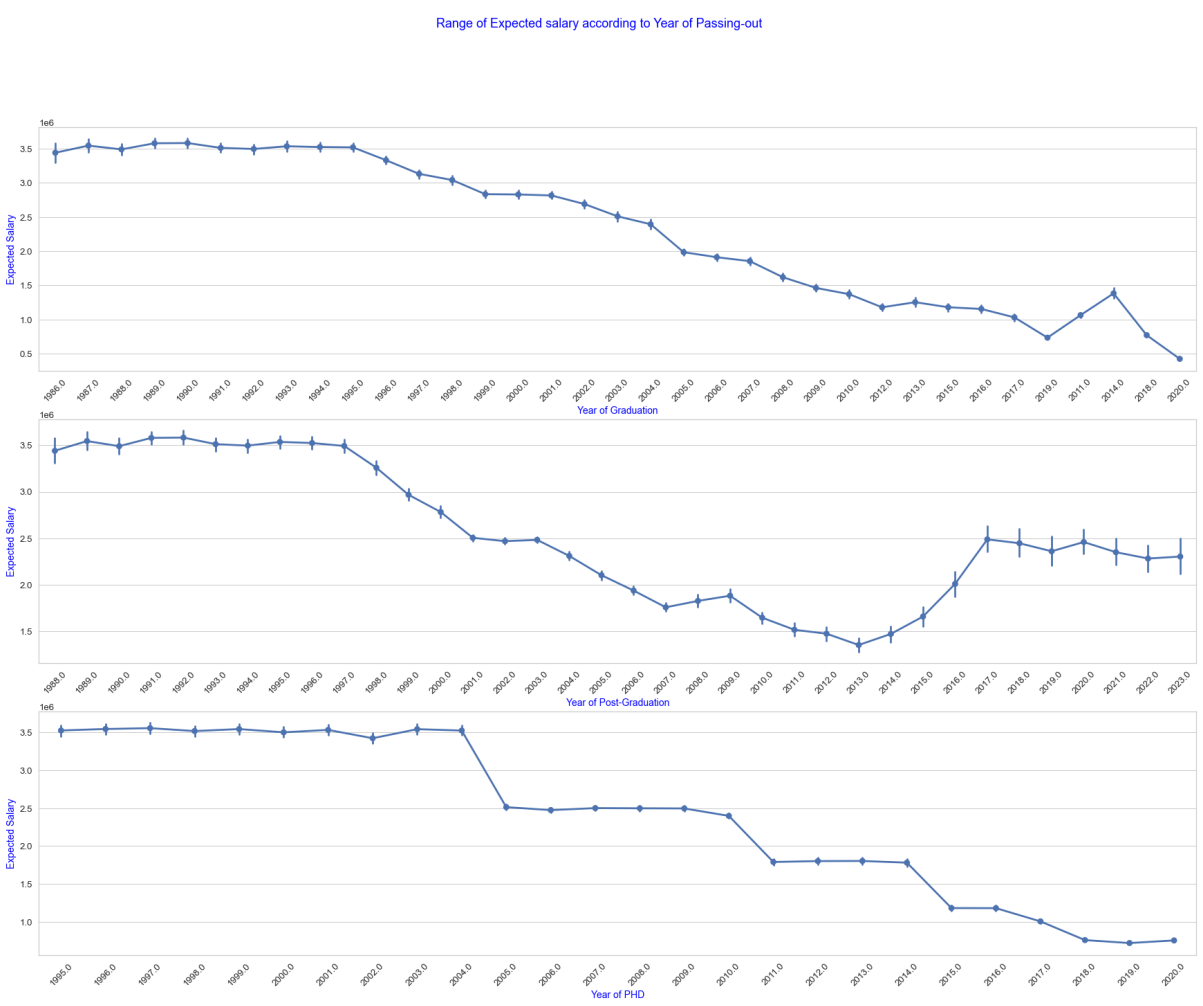
In [82]:
```
fig, ax = plt.subplots(figsize=(20,10))
sns.boxplot(x='Education',y='Expected_CTC',hue='International_degre
plt.title("Range of Expected_CTC by Education",color="blue",fontsiz
plt.xlabel("Education",color="blue",fontsize=14)
plt.ylabel("Expected_CTC",color="blue",fontsize=14)
plt.show()
```



In [60]:
```
# Plotting to see impact of year of pass-out on target variable:

sns.set_style('whitegrid')
plt.figure(figsize=(30,22))
plt.suptitle("Range of Expected salary according to Year of Passing
plt.subplot(3,1,1)
sns.pointplot(x=expected_ctc['Passing_Year_Of_Graduation'],y=expect
plt.ylabel("Expected Salary",color="blue",fontsize=14)
plt.xlabel("Year of Graduation",color="blue",fontsize=14)
plt.xticks(rotation=45)
plt.subplot(3,1,2)
sns.pointplot(x=expected_ctc['Passing_Year_Of_PG'],y=expected_ctc['
plt.ylabel("Expected Salary",color="blue",fontsize=14)
plt.xlabel("Year of Post-Graduation",color="blue",fontsize=14)
plt.xticks(rotation=45)
plt.subplot(3,1,3)
sns.pointplot(x=expected_ctc['Passing_Year_Of_PHD'],y=expected_ctc[
plt.ylabel("Expected Salary",color="blue",fontsize=14)
plt.xlabel("Year of PHD",color="blue",fontsize=14)
plt.xticks(rotation=45)
```

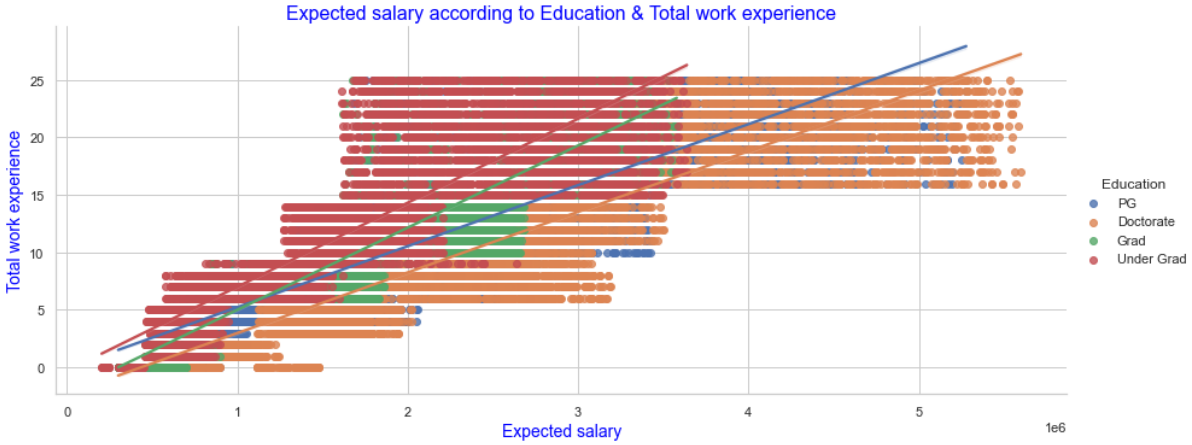
Out[60]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14
, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25]),
 [Text(0, 0, '1995.0'),
  Text(1, 0, '1996.0'),
```

```
Text(2, 0, '1997.0'),
Text(3, 0, '1998.0'),
Text(4, 0, '1999.0'),
Text(5, 0, '2000.0'),
Text(6, 0, '2001.0'),
Text(7, 0, '2002.0'),
Text(8, 0, '2003.0'),
Text(9, 0, '2004.0'),
Text(10, 0, '2005.0'),
Text(11, 0, '2006.0'),
Text(12, 0, '2007.0'),
Text(13, 0, '2008.0'),
Text(14, 0, '2009.0'),
Text(15, 0, '2010.0'),
Text(16, 0, '2011.0'),
Text(17, 0, '2012.0'),
Text(18, 0, '2013.0'),
Text(19, 0, '2014.0'),
Text(20, 0, '2015.0'),
Text(21, 0, '2016.0'),
Text(22, 0, '2017.0'),
Text(23, 0, '2018.0'),
Text(24, 0, '2019.0'),
Text(25, 0, '2020.0')])
```

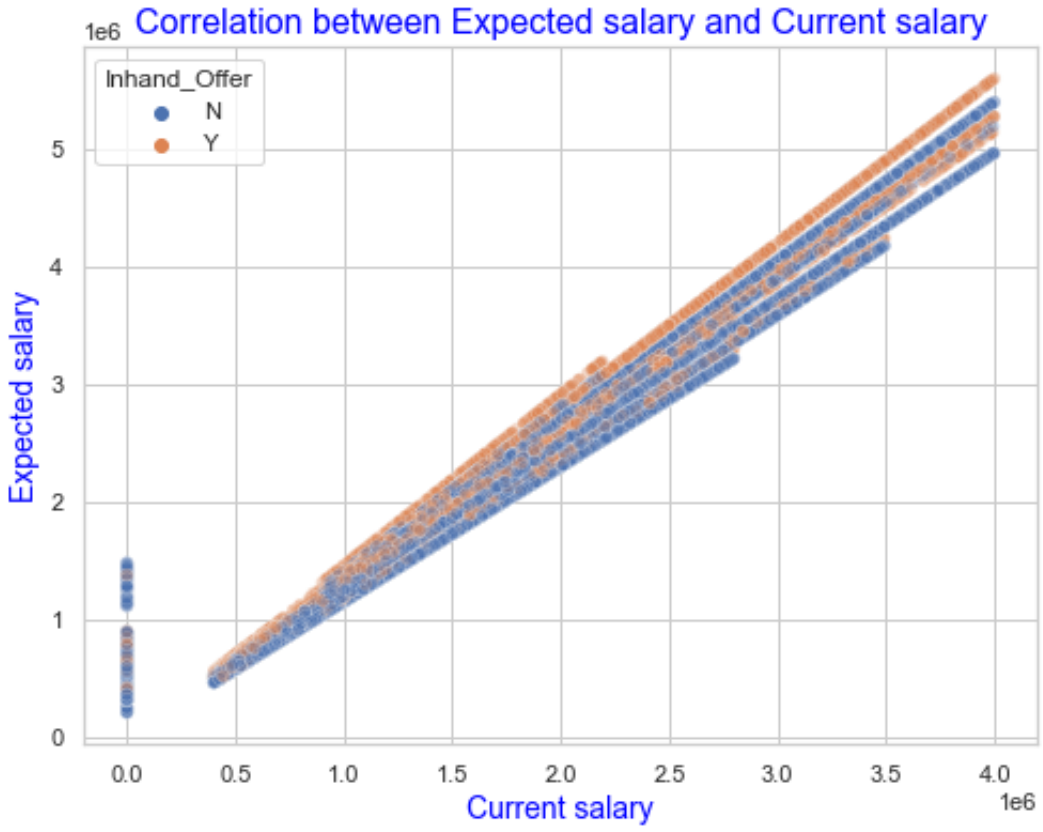


In [84]:
```
sns.lmplot(x="No_Of_Companies_worked", y="Expected_CTC", data=expec
plt.title("Expected salary according to No of companies worked & Pa
plt.ylabel("Expected Salary",color="blue",fontsize=14)
plt.xlabel("No of companies worked",color="blue",fontsize=14)
plt.show()
```

In [85]:
```
sns.lmplot(x="Expected_CTC",y="Total_Experience",data=expected_ctc,
plt.title("Expected salary according to Education & Total work expe
plt.ylabel("Total work experience",color="blue",fontsize=14)
plt.xlabel("Expected salary",color="blue",fontsize=14)
plt.show()
```



In [86]:
```
plt.figure(figsize=(8,6))

sns.scatterplot(x="Current_CTC", y="Expected_CTC", hue="Inhand_Offe
plt.title("Correlation between Expected salary and Current salary",
plt.xlabel("Current salary",color="blue",fontsize=14)
plt.ylabel("Expected salary",color="blue",fontsize=14)
plt.show()
```



In [87]:
```
# Profile of candidate with highest expected salary:

expected_ctc[expected_ctc['Expected_CTC']== expected_ctc['Expected_
```

Out[87]:

	Role	Designation	Education	Last_Appraisal_Rating	Total_Experience	Inhand_Of
22379	Financial Analyst	Marketing Manager	Doctorate	Key_Performer	17	

In [88]:
```
# Profile of candidate with lowest expected salary:

expected_ctc[expected_ctc['Expected_CTC']== expected_ctc['Expected_
```

Out[88]:

	Role	Designation	Education	Last_Appraisal_Rating	Total_Experience	Inhand_Offer
6788	NaN	NaN	Under Grad	NaN	0	N

In [89]:
```
#Checking for NaN values:

expected_ctc.isnull().values.any()
```

Out[89]:
```
True
```

In [90]:
```
check_nan_in_data=expected_ctc.isnull().sum()
print(check_nan_in_data)
```

```
Total_Experience                        0
Total_Experience_in_field_applied       0
Department                           2778
Role                                  963
Industry                              908
Organization                          908
Designation                          3129
Education                               0
Graduation_Specialization            6180
University_Grad                      6180
Passing_Year_Of_Graduation           6180
PG_Specialization                    7692
University_PG                        7692
Passing_Year_Of_PG                   7692
PHD_Specialization                  11881
University_PHD                      11881
Passing_Year_Of_PHD                 11881
Curent_Location                         0
Preferred_location                      0
Current_CTC                             0
Inhand_Offer                            0
Last_Appraisal_Rating                 908
No_Of_Companies_worked                  0
Number_of_Publications                  0
Certifications                          0
International_degree_any                0
Expected_CTC                            0
dtype: int64
```

In [ ]:
```
### Handling NaN values in categorical variables:

# Most NaN values are correspondent with rows where 'Total_experien
```

In [91]:
```
# Creating a subset of data where 'Total_experience' is 0:

zero_exp=expected_ctc[expected_ctc["Total_Experience"]==0]
zero_exp.head(25)
```

Out[91]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	Organi
0	0	0	NaN	NaN	NaN	
13	0	0	NaN	NaN	NaN	
140	0	0	NaN	NaN	NaN	
150	0	0	NaN	NaN	NaN	
176	0	0	NaN	NaN	NaN	
181	0	0	NaN	NaN	NaN	
206	0	0	NaN	NaN	NaN	
281	0	0	NaN	NaN	NaN	
311	0	0	NaN	NaN	NaN	
356	0	0	NaN	NaN	NaN	
392	0	0	NaN	NaN	NaN	
442	0	0	NaN	NaN	NaN	
446	0	0	NaN	NaN	NaN	
455	0	0	NaN	NaN	NaN	
488	0	0	NaN	NaN	NaN	
556	0	0	NaN	NaN	NaN	
588	0	0	NaN	NaN	NaN	
589	0	0	NaN	NaN	NaN	
622	0	0	NaN	NaN	NaN	
637	0	0	NaN	NaN	NaN	
664	0	0	NaN	NaN	NaN	
672	0	0	NaN	NaN	NaN	
709	0	0	NaN	NaN	NaN	
717	0	0	NaN	NaN	NaN	
734	0	0	NaN	NaN	NaN	

In [92]:
```
# In columns - ""Industry","Last_Appraisal_Rating","Organization","

cat=["Industry","Last_Appraisal_Rating","Organization","Department"

for column in expected_ctc[cat]:
    expected_ctc[column]=expected_ctc[column].fillna("None")
```

In [93]:
```
# Checking if NaN are replace correctly:

zero_exp=expected_ctc[expected_ctc["Total_Experience"]==0]
zero_exp.head(10)
```

Out[93]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	Organi
0	0	0	None	None	None	
13	0	0	None	None	None	
140	0	0	None	None	None	
150	0	0	None	None	None	
176	0	0	None	None	None	
181	0	0	None	None	None	
206	0	0	None	None	None	
281	0	0	None	None	None	
311	0	0	None	None	None	
356	0	0	None	None	None	

In [94]:
```
# Rechecking null values in the dataset:

expected_ctc.isnull().sum().sort_values()
```

Out[94]:
```
Total_Experience                          0
Certifications                            0
Number_of_Publications                    0
No_Of_Companies_worked                    0
Last_Appraisal_Rating                     0
Inhand_Offer                              0
Current_CTC                               0
Preferred_location                        0
Curent_Location                           0
International_degree_any                  0
Expected_CTC                              0
Education                                 0
Designation                               0
Organization                              0
Industry                                  0
Role                                      0
Department                                0
Total_Experience_in_field_applied         0
University_Grad                        6180
Graduation_Specialization              6180
Passing_Year_Of_Graduation             6180
PG_Specialization                      7692
University_PG                          7692
Passing_Year_Of_PG                     7692
PHD_Specialization                    11881
University_PHD                        11881
Passing_Year_Of_PHD                   11881
dtype: int64
```

In [95]:
```
expected_ctc['Department'].unique()
```

Out[95]:
```
array(['None', 'HR', 'Top Management', 'Banking', 'Sales', 'Engine
ering',
       'Others', 'Analytics/BI', 'Education', 'Marketing', 'Health
care',
       'IT-Software', 'Accounts'], dtype=object)
```

In [96]:
```
## Similarly in columns - 'Graduation_Specialization','University_G

cat=['Graduation_Specialization','University_Grad','Passing_Year_Of

for column in expected_ctc[cat]:
    expected_ctc[column]=expected_ctc[column].fillna("Not_Applicabl
```

In [97]:
```
# Checking if NaN are replaced correctly:

expected_ctc.head(10)
```

Out[97]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	Or
0	0	0	None	None	None	
1	23	14	HR	Consultant	Analytics	
2	21	12	Top Management	Consultant	Training	
3	15	8	Banking	Financial Analyst	Aviation	
4	10	5	Sales	Project Manager	Insurance	
5	16	3	Top Management	Area Sales Manager	Retail	
6	1	1	Engineering	Team Lead	FMCG	
7	19	11	Others	Analyst	Others	
8	8	7	Analytics/BI	Others	Telecom	
9	15	15	Analytics/BI	CEO	Telecom	

In [98]:
```
expected_ctc.isnull().sum()
```

Out[98]:
```
Total_Experience                      0
Total_Experience_in_field_applied     0
Department                            0
Role                                  0
Industry                              0
Organization                          0
Designation                           0
Education                             0
Graduation_Specialization             0
University_Grad                       0
Passing_Year_Of_Graduation            0
PG_Specialization                     0
University_PG                         0
Passing_Year_Of_PG                    0
PHD_Specialization                    0
University_PHD                        0
Passing_Year_Of_PHD                   0
Curent_Location                       0
Preferred_location                    0
Current_CTC                           0
Inhand_Offer                          0
Last_Appraisal_Rating                 0
No_Of_Companies_worked                0
Number_of_Publications                0
Certifications                        0
International_degree_any              0
Expected_CTC                          0
dtype: int64
```

In [ ]:
```
### Converting discrete categorical to discrete numerical variables
```

In [101]:
```
# Converting 'Inhand_Offer' to boolean values:

expected_ctc['Inhand_Offer'].replace(['N','Y'],[0,1],inplace=True )
```

In [102]:
```
expected_ctc['Inhand_Offer'].unique()
```

Out[102]:
```
array([0, 1])
```

In [ ]:
```
# There seem to be errors in the 'Education' column, since it is no
# Assuming that 'Education' has errors in it, we can create a new c
```

In [103]:
```
conditions=[
    (expected_ctc['Passing_Year_Of_PHD']=='Not_Applicable') & (expe
    (expected_ctc['Passing_Year_Of_PHD']=='Not_Applicable') & (expe
    (expected_ctc['Passing_Year_Of_PHD']=='Not_Applicable') & (expe
    (expected_ctc['Passing_Year_Of_PHD']!='Not_Applicable') & (expe
    ]
```

In [104]:
```
values=['Under-Grad','Graduate','Post-Grad','Doctorate']
```

In [105]:
```
expected_ctc['Edu_qualification']=np.select(conditions,values)
```

In [106]:
```
expected_ctc.head(25)
```

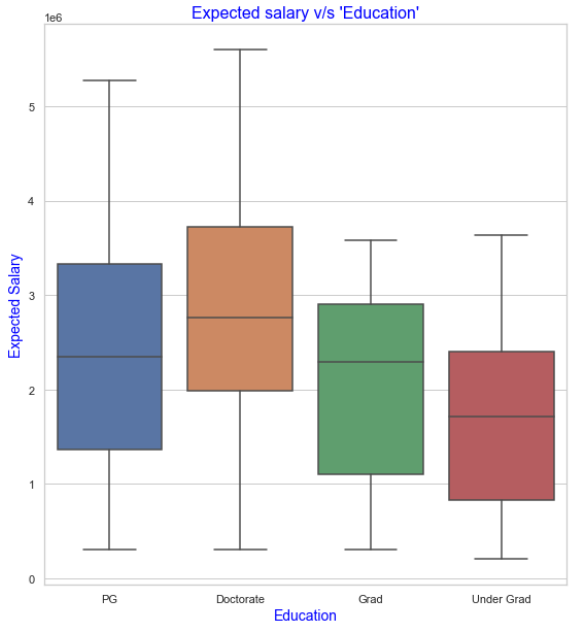
Out[106]:

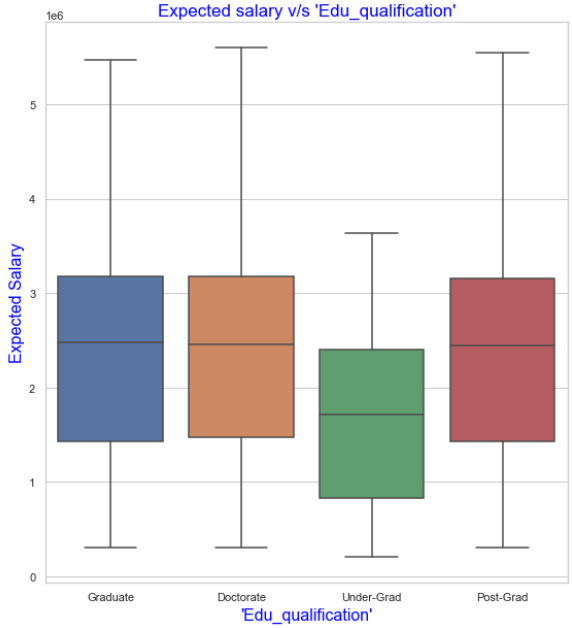
	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry
0	0	0	None	None	None
1	23	14	HR	Consultant	Analytics
2	21	12	Top Management	Consultant	Training
3	15	8	Banking	Financial Analyst	Aviation
4	10	5	Sales	Project Manager	Insurance
5	16	3	Top Management	Area Sales Manager	Retail
6	1	1	Engineering	Team Lead	FMCG
7	19	11	Others	Analyst	Others
8	8	7	Analytics/BI	Others	Telecom
9	15	15	Analytics/BI	CEO	Telecom
10	13	10	Education	Business Analyst	Automobile
11	7	1	Marketing	Sales Manager	FMCG
12	10	10	Others	Bio statistician	Automobile
13	0	0	None	None	None
14	12	9	Banking	Bio statistician	Telecom
15	20	15	Healthcare	Analyst	IT
16	4	4	Analytics/BI	Scientist	Analytics

17	21	7	Healthcare	Research Scientist	BFSI
18	14	9	Sales	Business Analyst	Telecom
19	8	3	Engineering	Consultant	Telecom
20	17	12	HR	Others	Training
21	7	6	Banking	Analyst	Training
22	22	6	Top Management	Consultant	BFSI
23	15	10	Sales	Head	Insurance
24	3	2	Banking	Associate	Aviation

In [110]:
```
# Checking to see the difference between the 2 columns by plotting


sns.set_style('whitegrid')
plt.figure(figsize=(20,10))
hue_order=expected_ctc['Education'].unique()
plt.subplot(1,2,1)
sns.boxplot(x=expected_ctc['Education'],y=expected_ctc['Expected_CT
plt.title("Expected salary v/s 'Education'",color="blue",fontsize=1
plt.ylabel("Expected Salary",color="blue",fontsize=14)
plt.xlabel("Education",color="blue",fontsize=14)
plt.subplot(1,2,2)
sns.boxplot(x=expected_ctc['Edu_qualification'],y=expected_ctc['Exp
plt.title("Expected salary v/s 'Edu_qualification'",color="blue",fo
plt.ylabel("Expected Salary",color="blue",fontsize=16)
plt.xlabel("'Edu_qualification'",color="blue",fontsize=16)
```

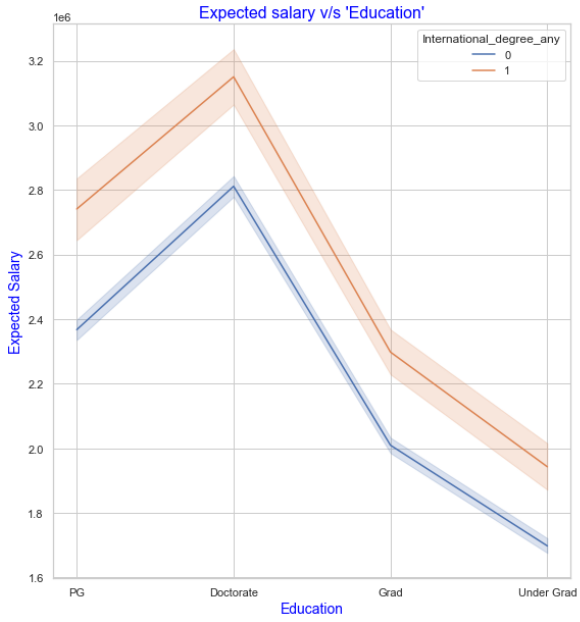
Out[110]: Text(0.5, 0, "'Edu_qualification'")

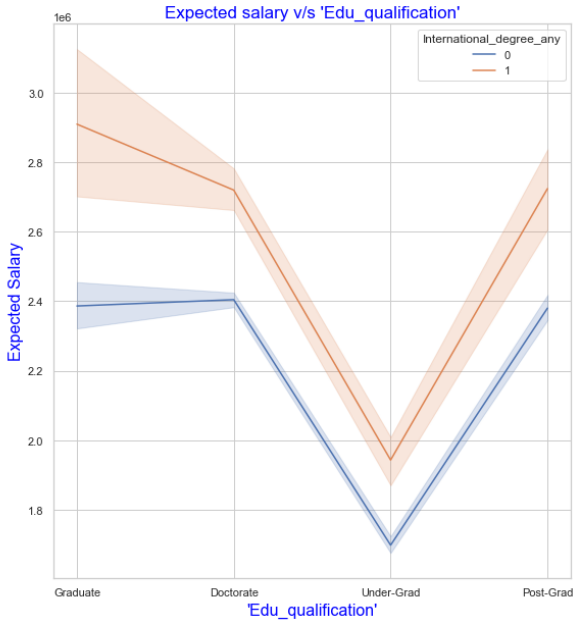




In [111]:
```
hecking to see the difference between the 2 columns by plotting ther


.set_style('whitegrid')
.figure(figsize=(20,10))
_order=expected_ctc['Education'].unique()
.subplot(1,2,1)
.lineplot(x=expected_ctc['Education'],y=expected_ctc['Expected_CTC']
.title("Expected salary v/s 'Education'",color="blue",fontsize=16)
.ylabel("Expected Salary",color="blue",fontsize=14)
.xlabel("Education",color="blue",fontsize=14)
.subplot(1,2,2)
.lineplot(x=expected_ctc['Edu_qualification'],y=expected_ctc['Expec
.title("Expected salary v/s 'Edu_qualification'",color="blue",fonts
.ylabel("Expected Salary",color="blue",fontsize=16)
.xlabel("'Edu_qualification'",color="blue",fontsize=16)
```

Out[111]: Text(0.5, 0, "'Edu_qualification'")





In [ ]: