In [1]:

```
import pandas as pd
```

In [2]:

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

In [3]:

```
df.head()
```

Out[3]:

	S.No.	College	Role	City type	Previous CTC	Previous job changes	Graduation marks	Exp (Months)	CTC
0	1	Tier 1	Manager	Non-Metro	55,523.00	3	66	19	71,406.58
1	2	Tier 2	Executive	Metro	57,081.00	1	84	18	68,005.87
2	3	Tier 2	Executive	Metro	60,347.00	2	52	28	76,764.02
3	4	Tier 3	Executive	Metro	49,010.00	2	81	33	82,092.39
4	5	Tier 3	Executive	Metro	57,879.00	4	74	32	73,878.10

In [4]:

```
df.tail()
```

Out[4]:

	S.No.	College	Role	City type	Previous CTC	Previous job changes	Graduation marks	Exp (Months)	CTC
1333	1334	Tier 3	Executive	Metro	59,661.00	4	68	50	69,712.40
1334	1335	Tier 1	Executive	Non-Metro	53,714.00	1	67	18	69,298.75
1335	1336	Tier 2	Executive	Non-Metro	61,957.00	1	47	18	66,397.77
1336	1337	Tier 1	Executive	Non-Metro	53,203.00	3	69	21	64,044.38
1337	1338	Tier 3	Manager	Non-Metro	51,820.00	1	47	61	83,346.06

In [5]:

```
df.shape
```

Out[5]:

```
(1338, 9)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['S.No.', 'College', 'Role', 'City type', 'Previous CTC',
       'Previous job changes', 'Graduation marks', 'Exp (Months)', 'CTC'],
      dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
S.No.                  0
College                0
Role                   0
City type              0
Previous CTC           0
Previous job changes   0
Graduation marks       0
Exp (Months)           0
CTC                    0
dtype: int64
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   S.No.                 1338 non-null   int64 
 1   College               1338 non-null   object
 2   Role                  1338 non-null   object
 3   City type             1338 non-null   object
 4   Previous CTC          1338 non-null   object
 5   Previous job changes  1338 non-null   int64 
 6   Graduation marks      1338 non-null   int64 
 7   Exp (Months)          1338 non-null   int64 
 8   CTC                   1338 non-null   object
dtypes: int64(4), object(5)
memory usage: 94.2+ KB
```

In [10]:

```
df.describe()
```

Out[10]:

	S.No.	Previous job changes	Graduation marks	Exp (Months)
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	669.500000	2.525411	59.890882	39.207025
std	386.391641	1.123502	14.894696	14.049960
min	1.000000	1.000000	35.000000	18.000000
25%	335.250000	2.000000	47.000000	27.000000
50%	669.500000	3.000000	60.000000	39.000000
75%	1003.750000	4.000000	73.000000	51.000000
max	1338.000000	4.000000	85.000000	64.000000

In [11]:

```
df = df.drop('S.No.', axis = 1)
```

In [12]:

```
df.nunique()
```

Out[12]:

```
College                   3
Role                      2
City type                 2
Previous CTC           1308
Previous job changes      4
Graduation marks         51
Exp (Months)             47
CTC                    1338
dtype: int64
```

In [13]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [14]:

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

Out[14]:

```
array(['Tier 1', 'Tier 2', 'Tier 3'], dtype=object)
```

In [15]:

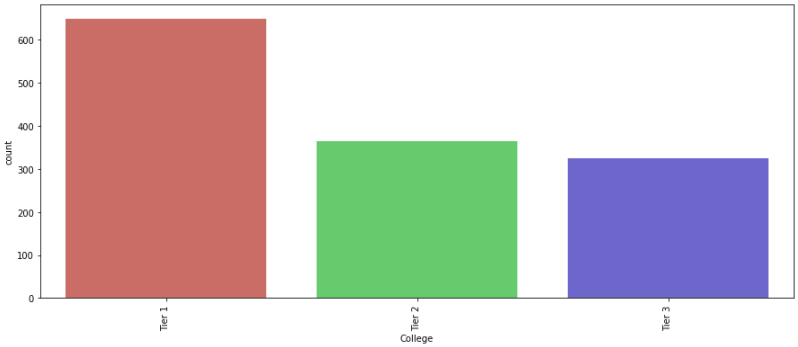
```
df['College'].value_counts()
```

Out[15]:

```
Tier 1    649
Tier 2    364
Tier 3    325
Name: College, dtype: int64
```

In [16]:

```
plt.figure(figsize=(15,6))
sns.countplot('College', data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



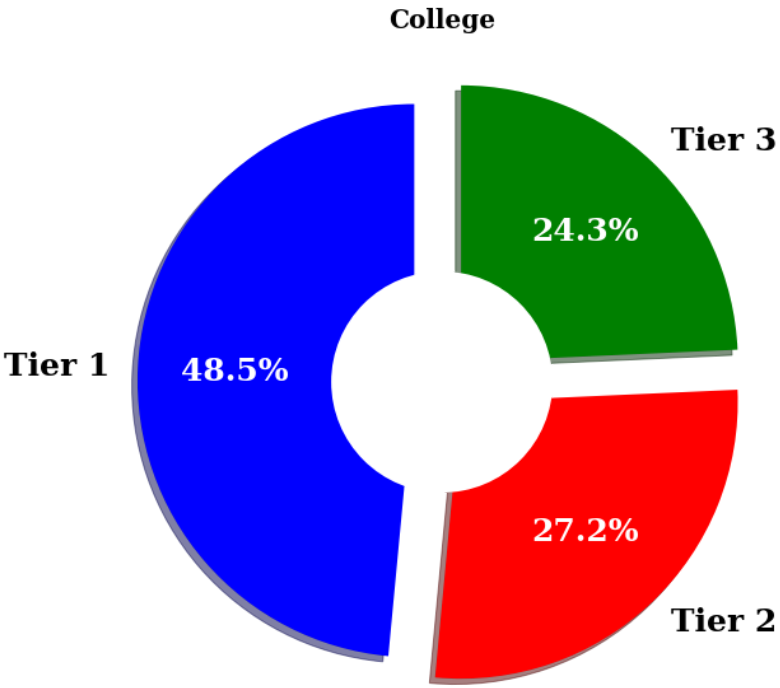
In [17]:

```
label_data = df['College'].value_counts()

explode = (0.1, 0.1, 0.1)
plt.figure(figsize=(14, 10))
patches, texts, pcts = plt.pie(label_data,
                               labels = label_data.index,
                               colors = ['blue', 'red', 'green'],
                               pctdistance = 0.65,
                               shadow = True,
                               startangle = 90,
                               explode = explode,
                               autopct = '%1.1f%%',
                               textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
plt.setp(pcts, color='white')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('College', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```



In [18]:

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

Out[18]:

```
array(['Manager', 'Executive'], dtype=object)
```

In [19]:

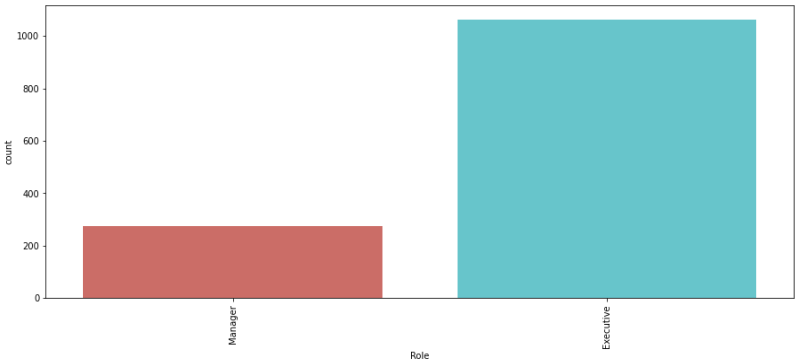
```
df['Role'].value_counts()
```

Out[19]:

```
Executive    1064
Manager       274
Name: Role, dtype: int64
```

In [20]:

```
plt.figure(figsize=(15,6))
sns.countplot('Role', data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



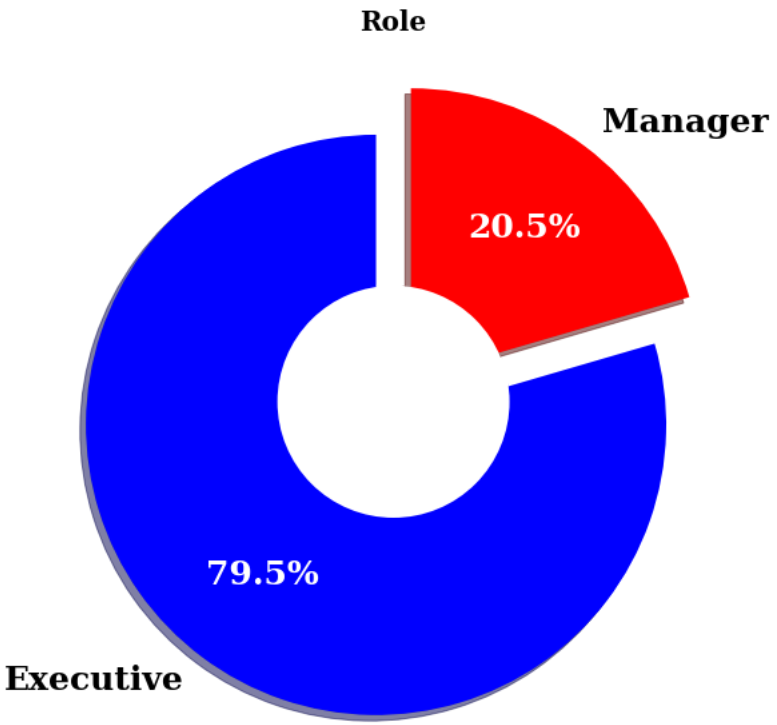
In [21]:

```
label_data = df['Role'].value_counts()

explode = (0.1, 0.1)
plt.figure(figsize=(14, 10))
patches, texts, pcts = plt.pie(label_data,
                               labels = label_data.index,
                               colors = ['blue', 'red'],
                               pctdistance = 0.65,
                               shadow = True,
                               startangle = 90,
                               explode = explode,
                               autopct = '%1.1f%%',
                               textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
plt.setp(pcts, color='white')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Role', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```



In [22]:

```
df['City type'].unique()
```

Out[22]:

```
array(['Non-Metro', 'Metro'], dtype=object)
```

In [23]:

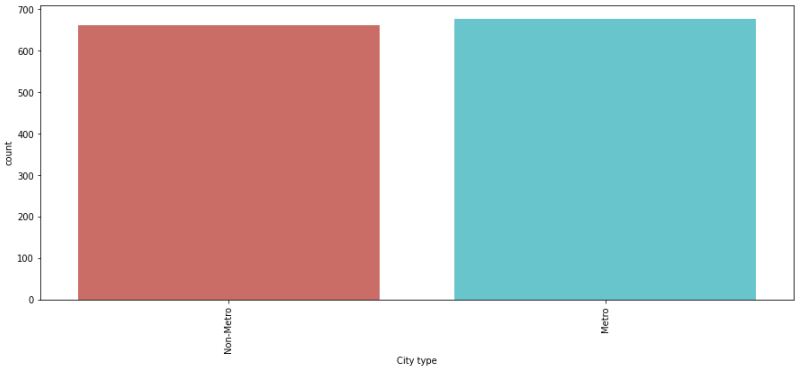
```
df['City type'].value_counts()
```

Out[23]:

```
Metro        676
Non-Metro    662
Name: City type, dtype: int64
```

In [24]:

```
plt.figure(figsize=(15,6))
sns.countplot('City type', data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



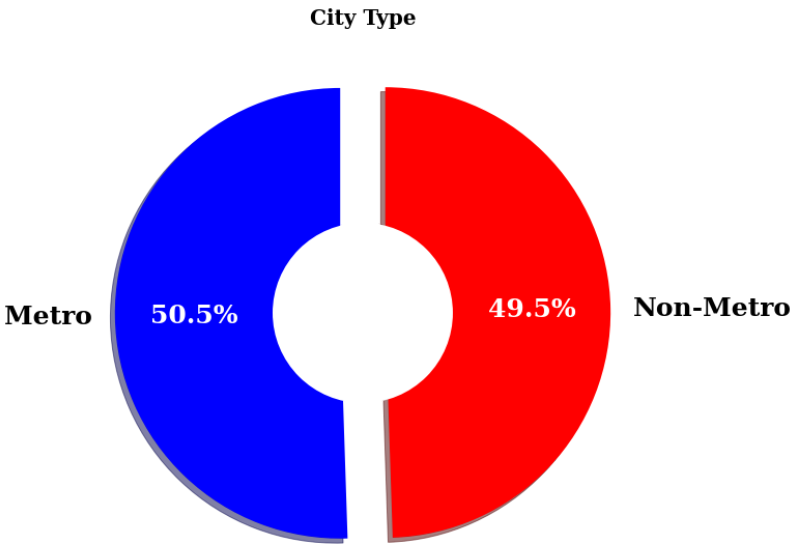
In [25]:

```
label_data = df['City type'].value_counts()

explode = (0.1, 0.1)
plt.figure(figsize=(14, 10))
patches, texts, pcts = plt.pie(label_data,
                               labels = label_data.index,
                               colors = ['blue', 'red'],
                               pctdistance = 0.65,
                               shadow = True,
                               startangle = 90,
                               explode = explode,
                               autopct = '%1.1f%%',
                               textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
plt.setp(pcts, color='white')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('City Type', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```



In [26]:

```
df['Previous job changes'].unique()
```

Out[26]:

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

In [27]:

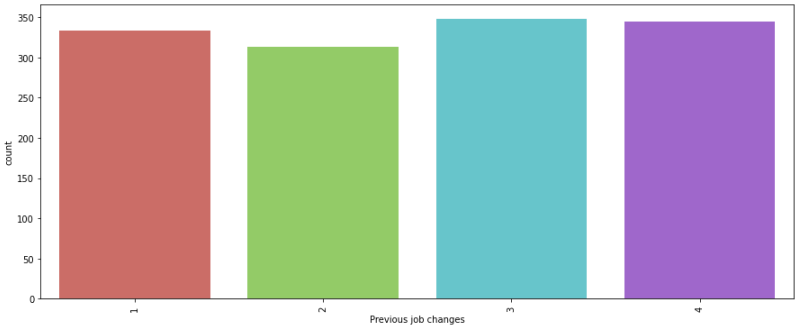
```
df['Previous job changes'].value_counts()
```

Out[27]:

```
3    348
4    344
1    333
2    313
Name: Previous job changes, dtype: int64
```

In [28]:

```
plt.figure(figsize=(15,6))
sns.countplot('Previous job changes', data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



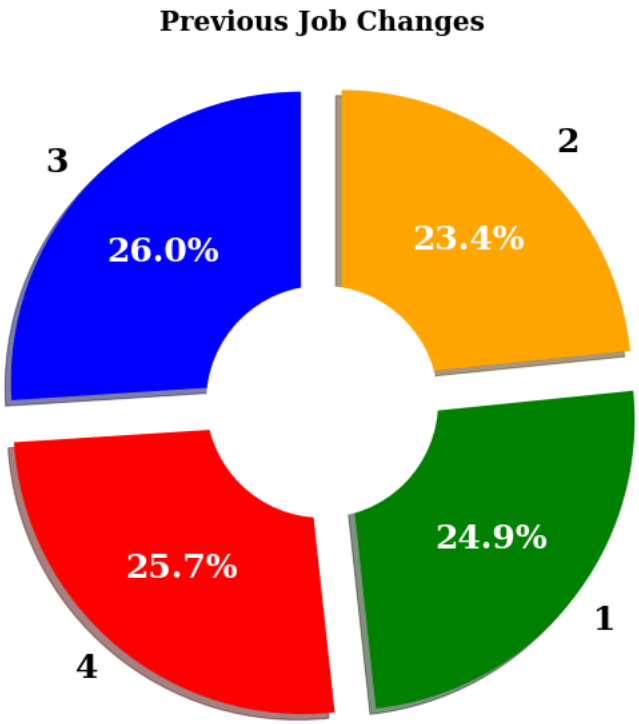
In [29]:

```
label_data = df['Previous job changes'].value_counts()

explode = (0.1, 0.1, 0.1, 0.1)
plt.figure(figsize=(14, 10))
patches, texts, pcts = plt.pie(label_data,
                               labels = label_data.index,
                               colors = ['blue', 'red', 'green', 'orange'],
                               pctdistance = 0.65,
                               shadow = True,
                               startangle = 90,
                               explode = explode,
                               autopct = '%1.1f%%',
                               textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
plt.setp(pcts, color='white')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Previous Job Changes', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```

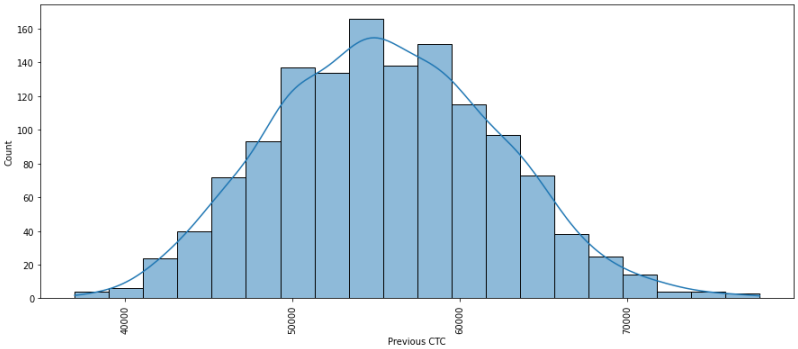


In [30]:

```
df['Previous CTC'] = df['Previous CTC'].str.replace(',', '')
df['CTC'] = df['CTC'].str.replace(',', '')
df['Previous CTC'] = df['Previous CTC'].astype(float)
df['CTC'] = df['CTC'].astype(float)
```

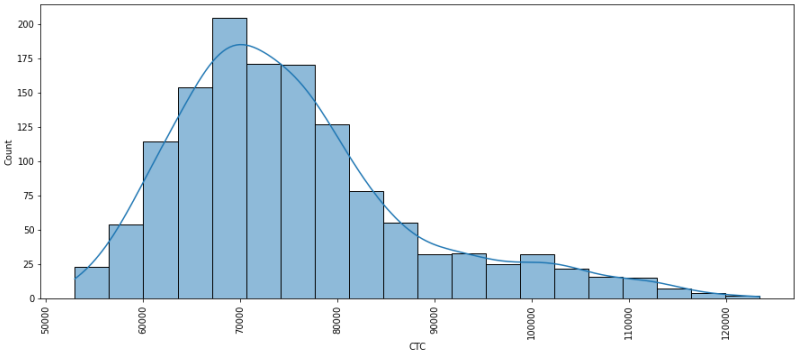
In [31]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['Previous CTC'], kde = True, bins = 20, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



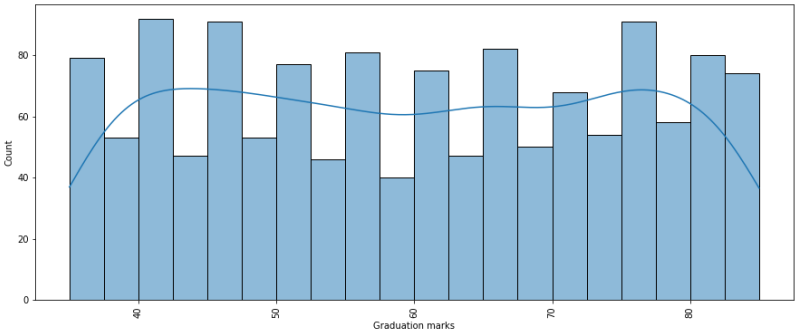
In [32]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['CTC'], kde = True, bins = 20, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



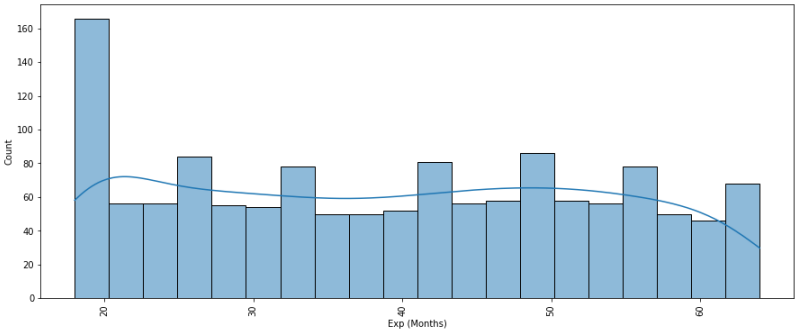
In [33]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['Graduation marks'], kde = True, bins = 20, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



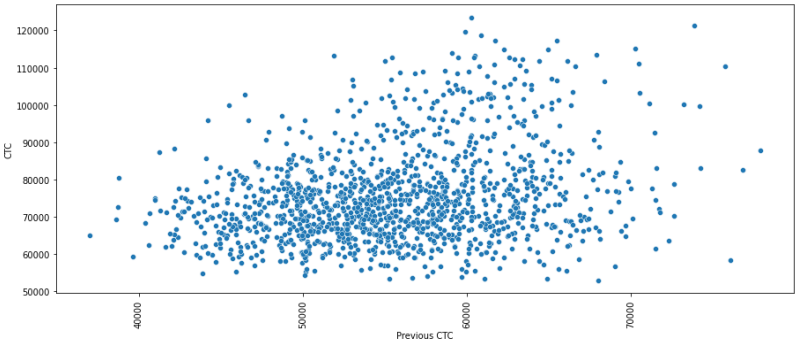
In [34]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['Exp (Months)'], kde = True, bins = 20, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



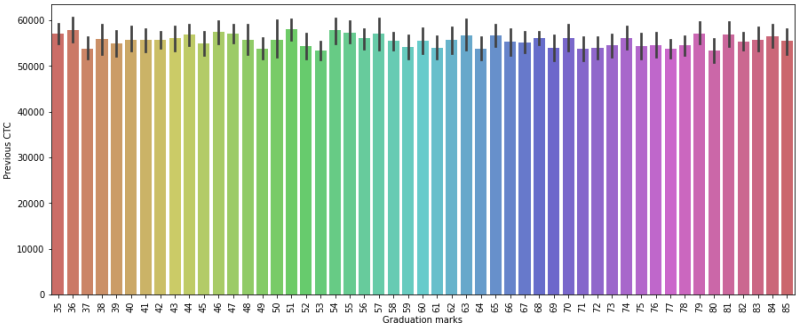
In [35]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(x= df['Previous CTC'], y = df['CTC'], data = df,
                palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



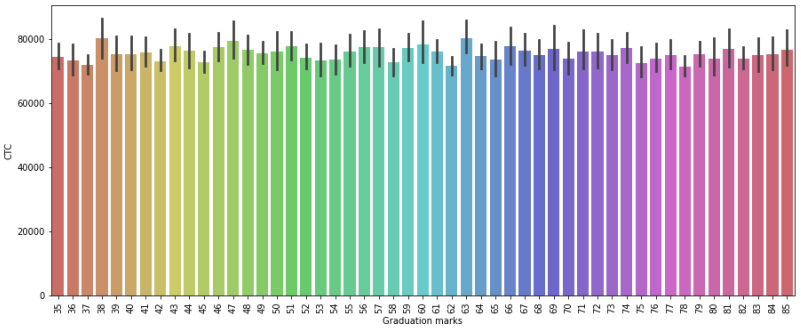
In [36]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['Previous CTC'], x = df['Graduation marks'], data = df,
                palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



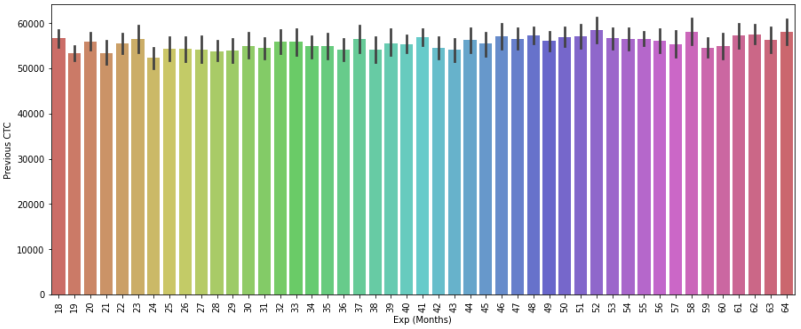
In [37]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['CTC'], x = df['Graduation marks'], data = df,
                palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



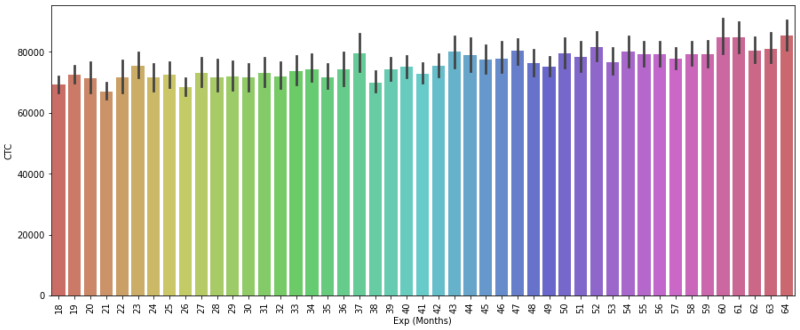
In [38]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['Previous CTC'], x = df['Exp (Months)'], data = df,
                palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [39]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['CTC'], x = df['Exp (Months)'], data = df,
                palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```

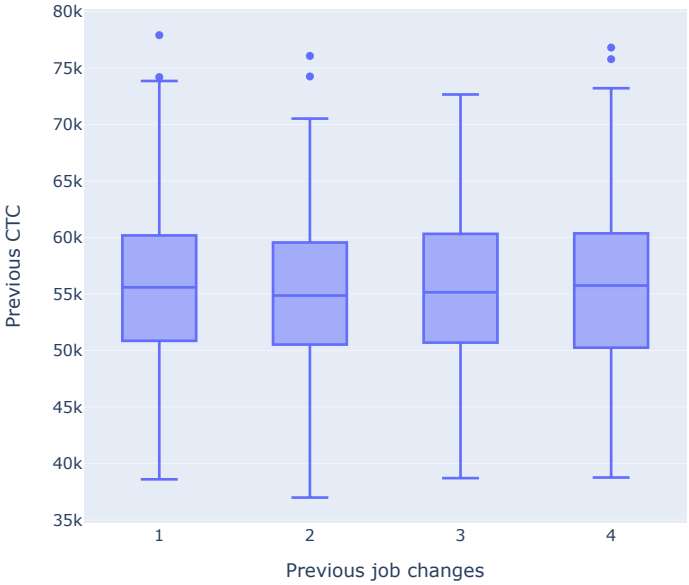


In [40]:

```
import plotly.express as px
```

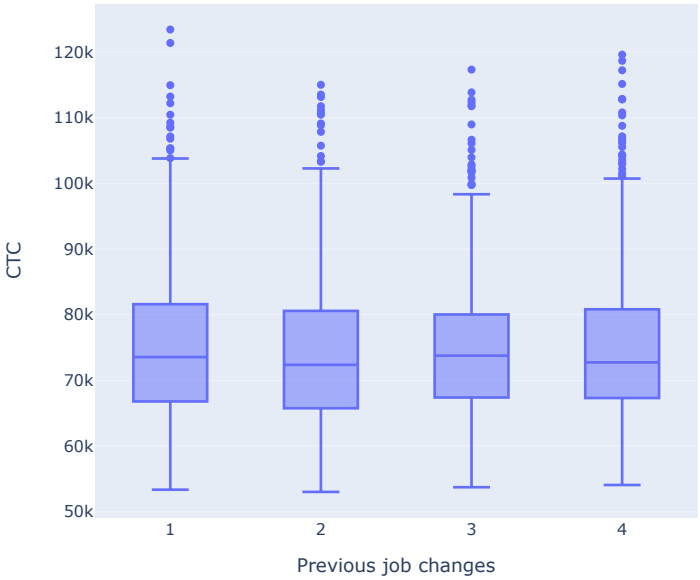
In [41]:

```
fig = px.box(df, x='Previous job changes', y='Previous CTC')
fig.show()
```



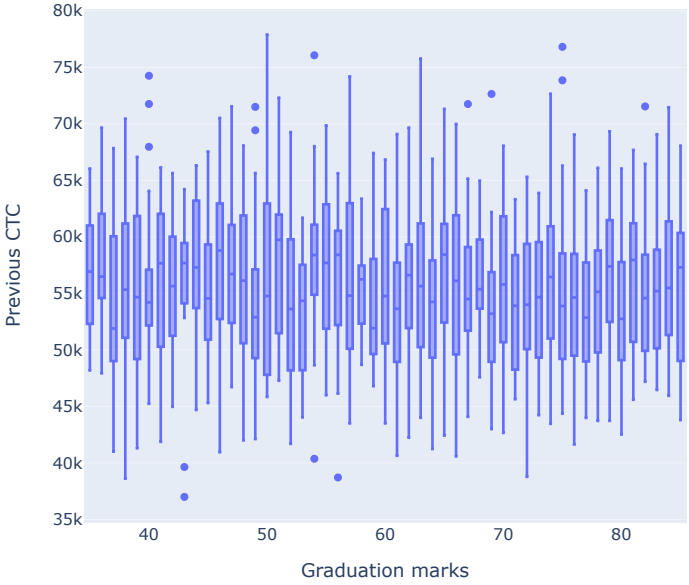
In [42]:

```
fig = px.box(df, x='Previous job changes', y='CTC')
fig.show()
```



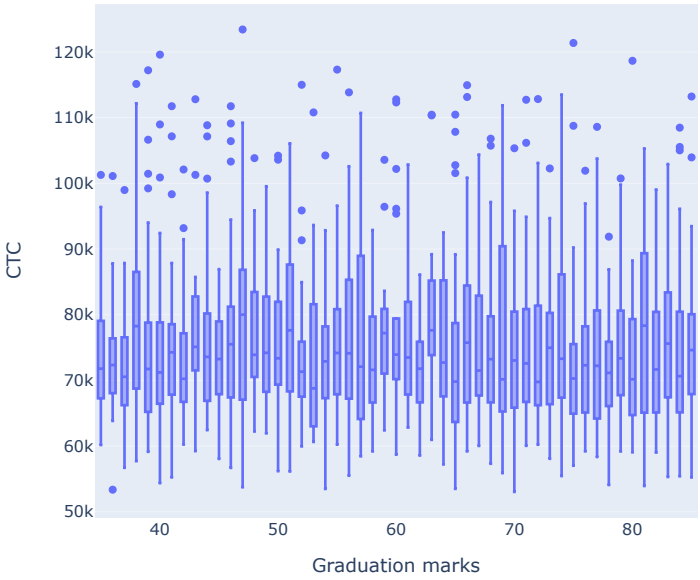
In [43]:

```
fig = px.box(df, x='Graduation marks', y='Previous CTC')
fig.show()
```



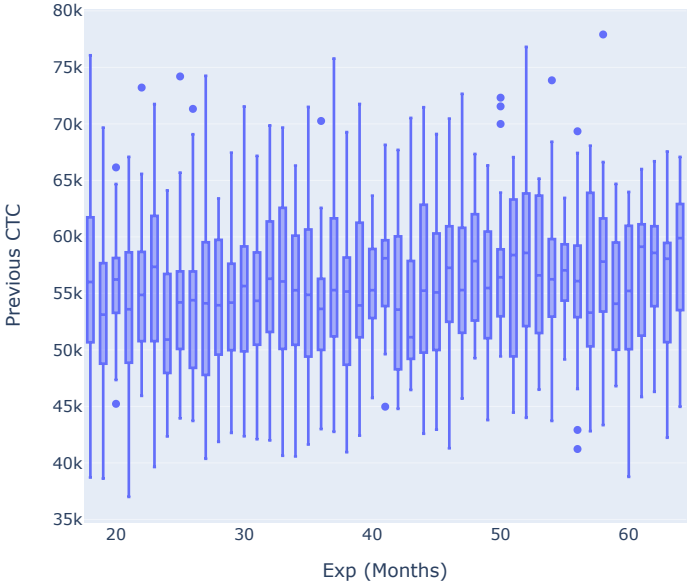
In [44]:

```
fig = px.box(df, x='Graduation marks', y='CTC')
fig.show()
```



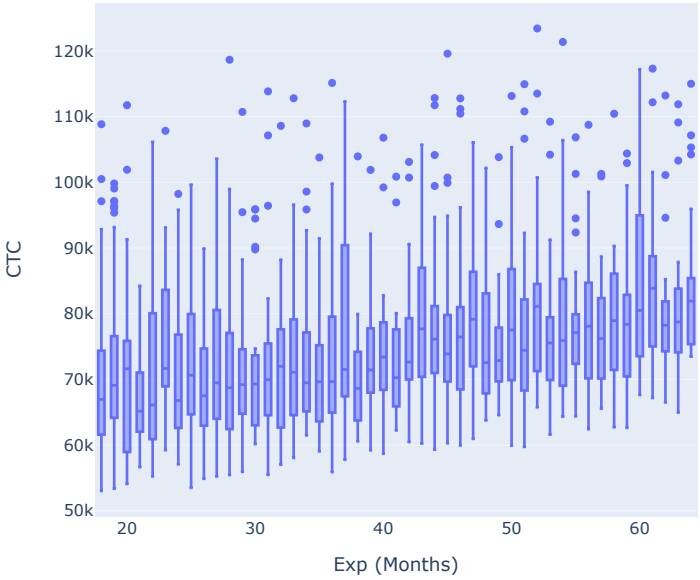
In [45]:

```
fig = px.box(df, x='Exp (Months)', y='Previous CTC')
fig.show()
```



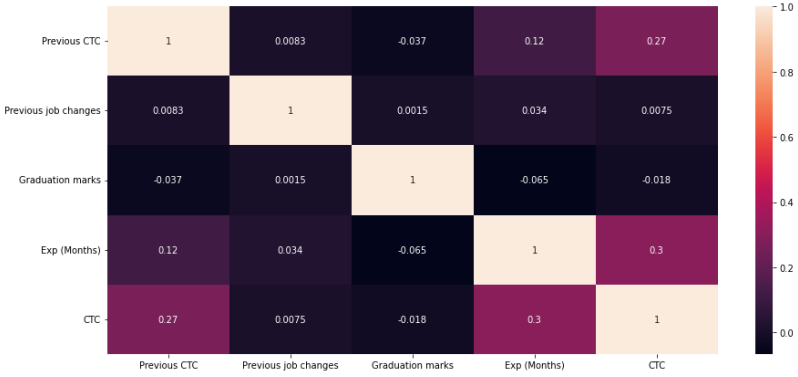
In [46]:

```
fig = px.box(df, x='Exp (Months)', y='CTC')
fig.show()
```



In [47]:

```
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(), annot=True)
plt.show()
```



In [48]:

```
cols = ['College', 'Role', 'City type']
```

In [49]:

```
for col in cols:
    one = pd.get_dummies(df[col],prefix=col)
    df = pd.concat([df,one],axis=1).drop(col,axis=1)
```

In [50]:

```
df
```

Out[50]:

	Previous CTC	Previous job changes	Graduation marks	Exp (Months)	CTC	College_Tier 1	College_Tier 2	College_Tier 3
0	55523.0	3	66	19	71406.58	1	0	0
1	57081.0	1	84	18	68005.87	0	1	0
2	60347.0	2	52	28	76764.02	0	1	0
3	49010.0	2	81	33	82092.39	0	0	1
4	57879.0	4	74	32	73878.10	0	0	1
...	...	...	...	...	...	...	...	...
1333	59661.0	4	68	50	69712.40	0	0	1
1334	53714.0	1	67	18	69298.75	1	0	0
1335	61957.0	1	47	18	66397.77	0	1	0
1336	53203.0	3	69	21	64044.38	1	0	0
1337	51820.0	1	47	61	83346.06	0	0	1

1338 rows × 12 columns

In [51]:

```
df.drop(['College_Tier 3', 'Role_Executive','City type_Non-Metro'], axis=1, inplace=True)
```

In [52]:

```
df
```

Out[52]:

	Previous CTC	Previous job changes	Graduation marks	Exp (Months)	CTC	College_Tier 1	College_Tier 2	Role_Manag
0	55523.0	3	66	19	71406.58	1	0	
1	57081.0	1	84	18	68005.87	0	1	
2	60347.0	2	52	28	76764.02	0	1	
3	49010.0	2	81	33	82092.39	0	0	
4	57879.0	4	74	32	73878.10	0	0	
...	...	...	...	...	...	...	...	
1333	59661.0	4	68	50	69712.40	0	0	
1334	53714.0	1	67	18	69298.75	1	0	
1335	61957.0	1	47	18	66397.77	0	1	
1336	53203.0	3	69	21	64044.38	1	0	
1337	51820.0	1	47	61	83346.06	0	0	

1338 rows × 9 columns

In [53]:

```
x = df.drop('CTC', axis=1)
y = df['CTC']
```

In [54]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.30)
```

In [55]:

```
from sklearn.linear_model import LinearRegression
```

In [56]:

```
lr = LinearRegression()
```

In [57]:

```
lr.fit(X_train,y_train)
```

Out[57]:

LinearRegression

LinearRegression()

In [58]:

```
y_pred = lr.predict(X_test)
```

In [59]:

```
from sklearn.metrics import mean_squared_error
```

In [60]:

```
RMSE_lr = mean_squared_error(y_test, y_pred, squared = False)
RMSE_lr
```

Out[60]:

```
7574.324445494704
```

In [61]:

```
from sklearn.model_selection import cross_val_score
```

In [62]:

```
cross_val_score(lr, X_train, y_train, cv=10)
```

Out[62]:

```
array([0.55212149, 0.53167995, 0.69645457, 0.47365687, 0.54144254,
       0.52123669, 0.59718042, 0.63818445, 0.71317137, 0.58601787])
```

In [63]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [64]:

```
dt = DecisionTreeRegressor(random_state=0)
```

In [65]:

```
dt.fit(X_train, y_train)
```

Out[65]:

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

In [66]:

```
y_pred = dt.predict(X_test)
```

In [67]:

```
RMSE_dt = mean_squared_error(y_test, y_pred, squared = False)
RMSE_dt
```

Out[67]:

```
10586.634234772882
```

In [68]:

```
cross_val_score(dt, X_train, y_train, cv=10)
```

Out[68]:

```
array([0.12533484, 0.28814434, 0.48973731, 0.17480582, 0.05752311,
       0.12373591, 0.33239776, 0.38863548, 0.52126077, 0.34205437])
```

In [69]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [70]:

```
rf = RandomForestRegressor(max_depth=2, random_state=0)
```

In [71]:

```
rf.fit(X_train, y_train)
```

Out[71]:

```
RandomForestRegressor
RandomForestRegressor(max_depth=2, random_state=0)
```

In [72]:

```
y_pred = rf.predict(X_test)
```

In [73]:

```
RMSE_rf = mean_squared_error(y_test, y_pred, squared = False)
RMSE_rf
```

Out[73]:

```
7733.97875142321
```

In [74]:

```
cross_val_score(rf, X_train, y_train, cv=10)
```

Out[74]:

```
array([0.46766987, 0.50678706, 0.67517662, 0.47267458, 0.56276318,
       0.35167094, 0.55308359, 0.57442213, 0.71419994, 0.6434533 ])
```

In [75]:

```
from sklearn.model_selection import GridSearchCV

# Define the grid of hyperparameters to search
param_grid = {'fit_intercept': [True, False],
              'normalize': [True, False]}

# Create a linear regression model
lin_reg = LinearRegression()

# Create a grid search object
grid_search = GridSearchCV(lin_reg, param_grid, cv=5)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_fit_intercept = grid_search.best_params_['fit_intercept']
best_normalize = grid_search.best_params_['normalize']

# Train a linear regression model with the best hyperparameters
lin_reg_best = LinearRegression(fit_intercept=best_fit_intercept,
                                normalize=best_normalize)
lin_reg_best.fit(X_train, y_train)
```

Out[75]:

```
LinearRegression(normalize=False)
```

In [76]:

```
y_pred = lin_reg_best.predict(X_test)
```

In [77]:

```
RMSE_lr = mean_squared_error(y_test, y_pred, squared = False)
RMSE_lr
```

Out[77]:

```
7574.324445494704
```

In [78]:

```
# Define the grid of hyperparameters to search
param_grid = {'max_depth': [1, 2, 3, 4, 5],
              'min_samples_split': [2, 3, 4]}

# Create a decision tree regression model
dtr = DecisionTreeRegressor()

# Create a grid search object
grid_search = GridSearchCV(dtr, param_grid, cv=5)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_max_depth = grid_search.best_params_['max_depth']
best_min_samples_split = grid_search.best_params_['min_samples_split']

# Train a decision tree regression model with the best hyperparameters
dtr_best = DecisionTreeRegressor(max_depth=best_max_depth,
                                 min_samples_split=best_min_samples_split)
dtr_best.fit(X_train, y_train)
```

Out[78]:

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=4)
```

In [79]:

```
y_pred = dtr_best.predict(X_test)
```

In [80]:

```
RMSE_dt = mean_squared_error(y_test, y_pred, squared = False)
RMSE_dt
```

Out[80]:

```
7361.444145934613
```

In [81]:

```
# Define the grid of hyperparameters to search
param_grid = {'n_estimators': [10, 50, 100],
              'max_depth': [1, 2, 3, 4, 5]}

# Create a random forest regression model
rfr = RandomForestRegressor()

# Create a grid search object
grid_search = GridSearchCV(rfr, param_grid, cv=5)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_n_estimators = grid_search.best_params_['n_estimators']
best_max_depth = grid_search.best_params_['max_depth']

# Train a random forest regression model with the best hyperparameters
rfr_best = RandomForestRegressor(n_estimators=best_n_estimators,
                                 max_depth=best_max_depth)
rfr_best.fit(X_train, y_train)
```

Out[81]:

```
RandomForestRegressor
RandomForestRegressor(max_depth=4, n_estimators=10)
```

In [82]:

```
y_pred = rfr_best.predict(X_test)
```

In [83]:

```
RMSE_rf = mean_squared_error(y_test, y_pred, squared = False)
RMSE_rf
```

Out[83]:

```
7061.411651886914
```