In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv ('hr_employee.csv')
df.head(5)
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employee
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

5 rows × 35 columns

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [4]:
```
df.columns.unique()
```

Out[4]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [5]:
```
df.columns.nunique()
```

Out[5]:
```
35
```

In [6]:
```
df.shape

# there are 1470 rows and 35 columns in the given dataset.
```

Out[6]:
```
(1470, 35)
```

In [7]:
```
column = ['age', 'attrition', 'business travel', 'daily rate', 'department',
       'distance from home', 'education', 'education field', 'employeecount',
       'employee number', 'environment satisfaction', 'gender', 'hourly rate',
       'job involvement', 'job level', 'job role', 'job satisfaction',
       'marital status', 'monthly income', 'monthly rate', 'num companies worked',
       'over18', 'over time', 'percent salary hike', 'performance rating',
       'relationship satisfaction', 'standard hours', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role', 'years since last promotion',
       'years with currmanager']
```

In [8]:
```
df.columns=column
# coverting all column names into same small character, thic would be helpful for future other
```

In [9]:
```
df.head(2)
# all column names are converted successfully into small characters.
```

Out[9]:

	age	attrition	business travel	daily rate	department	distance from home	education	education field	employeecount	employee number	...
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...

2 rows × 35 columns

In [10]:
```
df.columns.unique()
```

Out[10]:
```
Index(['age', 'attrition', 'business travel', 'daily rate', 'department',
       'distance from home', 'education', 'education field', 'employeecount',
       'employee number', 'environment satisfaction', 'gender', 'hourly rate',
       'job involvement', 'job level', 'job role', 'job satisfaction',
       'marital status', 'monthly income', 'monthly rate',
       'num companies worked', 'over18', 'over time', 'percent salary hike',
       'performance rating', 'relationship satisfaction', 'standard hours',
       'stock option Level', 'total working years', 'training times last year',
       'work life balance', 'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

In [11]:
```
df.dtypes
# most of the columns are in 'int64', except- 9 columns.
# out of 35 only 9 columns are in 'object' datatype
#  rest of the 26 columns are of 'inte64' datatype.
```

Out[11]:
```
age                          int64
attrition                   object
business travel             object
daily rate                   int64
department                  object
distance from home           int64
education                    int64
education field             object
employeecount                int64
employee number              int64
environment satisfaction     int64
gender                      object
hourly rate                  int64
job involvement              int64
job level                    int64
job role                    object
job satisfaction             int64
marital status              object
monthly income               int64
monthly rate                 int64
num companies worked         int64
over18                      object
over time                   object
percent salary hike          int64
performance rating           int64
relationship satisfaction    int64
standard hours               int64
stock option Level           int64
total working years          int64
training times last year     int64
work life balance            int64
years at company             int64
years in current role        int64
years since last promotion   int64
years with currmanager       int64
dtype: object
```

In [12]:
```
df.info()
#  here we can see that
# 1) total number for columns present : 35
# 2) total number of rows presnet : 1,470
# 3) total "data types present in data set" : 4 (i.e int64(26), object(9)")
#  (object)- 'attrition' 'business level' 'department' 'education field' 'gender' 'job role' 'm
# 4)NULL VALUES are may not be present in our dataset.
#    because there is no difference between the total count and present values.
#  5) So we can also to check 'null values' & 'whitespaces' in our dataset by futher different
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   age                        1470 non-null   int64 
 1   attrition                  1470 non-null   object
 2   business travel            1470 non-null   object
 3   daily rate                 1470 non-null   int64 
 4   department                 1470 non-null   object
 5   distance from home         1470 non-null   int64 
 6   education                  1470 non-null   int64 
 7   education field            1470 non-null   object
 8   employeecount              1470 non-null   int64 
 9   employee number            1470 non-null   int64 
 10  environment satisfaction   1470 non-null   int64 
 11  gender                     1470 non-null   object
 12  hourly rate                1470 non-null   int64 
 13  job involvement            1470 non-null   int64 
 14  job level                  1470 non-null   int64 
 15  job role                   1470 non-null   object
 16  job satisfaction           1470 non-null   int64 
 17  marital status             1470 non-null   object
 18  monthly income             1470 non-null   int64 
 19  monthly rate               1470 non-null   int64 
 20  num companies worked       1470 non-null   int64 
 21  over18                     1470 non-null   object
 22  over time                  1470 non-null   object
 23  percent salary hike        1470 non-null   int64 
 24  performance rating         1470 non-null   int64 
 25  relationship satisfaction  1470 non-null   int64 
 26  standard hours             1470 non-null   int64 
 27  stock option Level         1470 non-null   int64 
 28  total working years        1470 non-null   int64 
 29  training times last year   1470 non-null   int64 
 30  work life balance          1470 non-null   int64 
 31  years at company           1470 non-null   int64 
 32  years in current role      1470 non-null   int64 
 33  years since last promotion 1470 non-null   int64 
 34  years with currmanager     1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [ ]:
```
```

CHECKING NULL VALUES

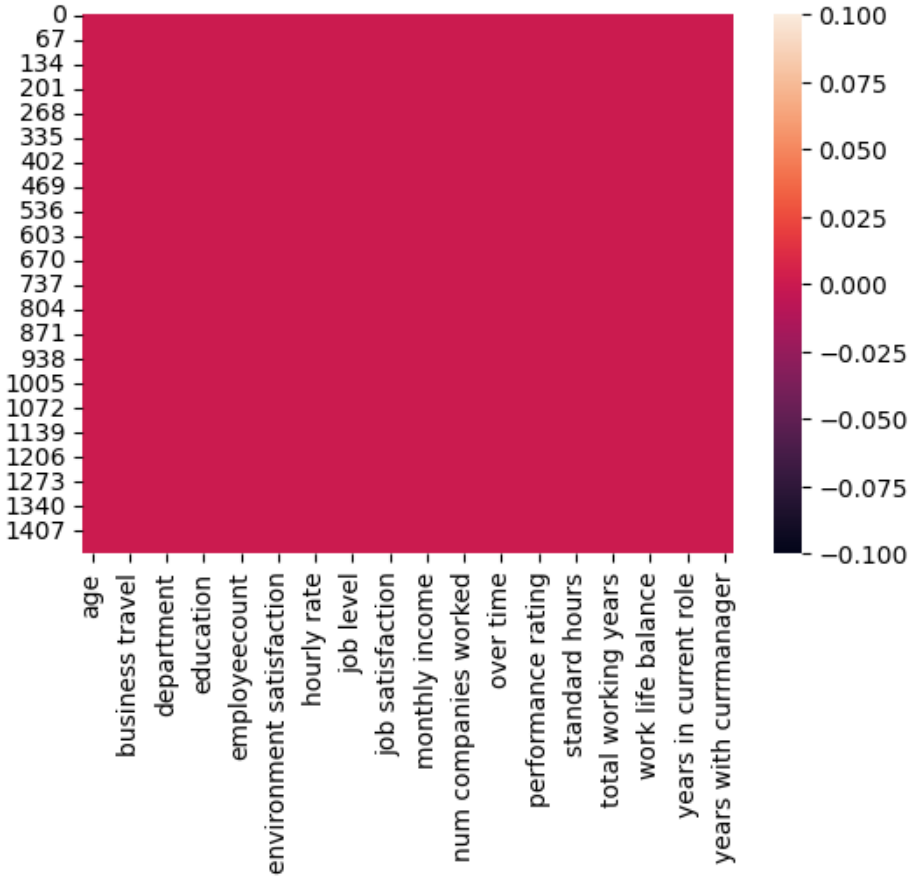
==========

In [13]:
```
df.isnull().sum()
#  here by following, we can clearly says that , there is NO NULL value is present in out datas
#  all the object columns are in correct formate that means , there are no white spaces in the
```

Out[13]:
```
age                          0
attrition                    0
business travel              0
daily rate                   0
department                   0
distance from home           0
education                    0
education field              0
employeecount                0
employee number              0
environment satisfaction     0
gender                       0
hourly rate                  0
job involvement              0
job level                    0
job role                     0
job satisfaction             0
marital status               0
monthly income               0
monthly rate                 0
num companies worked         0
over18                       0
over time                    0
percent salary hike          0
performance rating           0
relationship satisfaction    0
standard hours               0
stock option Level           0
total working years          0
training times last year     0
work life balance            0
years at company             0
years in current role        0
years since last promotion   0
years with currmanager       0
dtype: int64
```

In [14]:
```
plt.figure(figsize=(6,4))
sns.heatmap(df.isnull())

# here with the help of heatmap also we can see that there is not any presence of null values
```

Out[14]: <AxesSubplot:>



In [ ]:

CHECKING UNIQUE VALUES & PERFORMING UNIVARIATE ANALYSIS FOR EACH COLUMN

==========

In [15]:
```
df.shape
```

Out[15]: (1470, 35)

In [16]:
```
df.nunique()
#  here we can find the different number of unique values in each column.
# as we know that there are total 1470 rows are present in the given dataset.
# and here we can see that the column which consist very less number of unique value is =..
#  = attrition, business level, department, education, education field, employee count, enviorr
#  ... job involvement, job level, job role, job satisfaction, marital status, num companies wo
# .... performance rating, relationship satisfaction, standard hours, stock option level, trair
# ... so we can say that the above mentioned columns are 'Categorical Columns'
# here we can see that 'date' & 'average price' is also having very less unique numbers as comp
```

Out[16]:
```
age                          43
attrition                     2
business travel               3
daily rate                  886
department                    3
distance from home           29
education                     5
education field               6
employeecount                 1
employee number            1470
environment satisfaction      4
gender                        2
hourly rate                  71
job involvement               4
job level                     5
job role                      9
job satisfaction              4
marital status                3
monthly income             1349
monthly rate               1427
num companies worked         10
over18                        1
over time                     2
percent salary hike          15
performance rating            2
relationship satisfaction     4
standard hours                1
stock option Level            4
total working years          40
training times last year      7
work life balance             4
years at company             37
years in current role        19
years since last promotion   16
years with currmanager       18
dtype: int64
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['age', 'attrition', 'business travel', 'daily rate', 'department',
       'distance from home', 'education', 'education field', 'employeecount',
       'employee number', 'environment satisfaction', 'gender', 'hourly rate',
       'job involvement', 'job level', 'job role', 'job satisfaction',
       'marital status', 'monthly income', 'monthly rate',
       'num companies worked', 'over18', 'over time', 'percent salary hike',
       'performance rating', 'relationship satisfaction', 'standard hours',
       'stock option Level', 'total working years', 'training times last year',
       'work life balance', 'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

In [ ]:
```
```

In [18]:
```
# 1) Analysing AGE COLUMN ===>>
```

In [19]:
```
df['age'].unique()
```

Out[19]:
```
array([41, 49, 37, 33, 27, 32, 59, 30, 38, 36, 35, 29, 31, 34, 28, 22, 53,
       24, 21, 42, 44, 46, 39, 43, 50, 26, 48, 55, 45, 56, 23, 51, 40, 54,
       58, 20, 25, 19, 57, 52, 47, 18, 60], dtype=int64)
```

In [20]:
```
df['age'].nunique()
# there are 43 unique values are present the above mentioned column.
```

Out[20]:
```
43
```

In [21]:
```
df['age'].value_counts()
```

Out[21]:
```
35    78
34    77
36    69
31    69
29    68
32    61
30    60
33    58
38    58
40    57
37    50
27    48
28    48
42    46
39    42
45    41
41    40
26    39
44    33
46    33
43    32
50    30
25    26
24    26
49    24
47    24
55    22
51    19
53    19
48    19
54    18
52    18
22    16
56    14
23    14
58    14
21    13
20    11
59    10
19     9
18     8
60     5
57     4
Name: age, dtype: int64
```

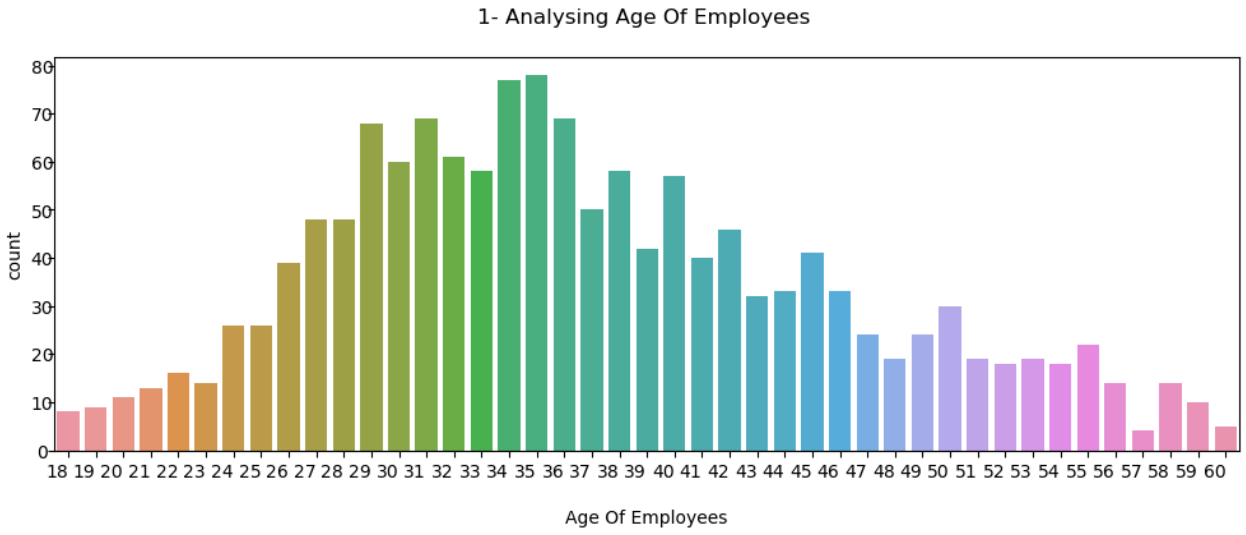
In [22]:
```
df['age'].min()
#  the minimum age present in the mentioned column. 18 year
```

Out[22]: 18

In [23]:
```
df['age'].max()
#  the Maximum age of the employee in the menitoned column is 60 Years
```

Out[23]: 60

In [24]:
```
plt.figure(figsize = (12,4), facecolor='white')
plt.title('\n1- Analysing Age Of Employees \n')
sns.countplot(x='age', data=df)
plt.xlabel('\nAge Of Employees', fontsize = 10)
plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below Countplot we can find that ,
#  The Minimum Age of Employees is = 18 years
#  The Higher Age of Employees is = 60 years
#  & most of the average Age of Emplyees are present in between (22 - 50 years)
```



In [ ]:
```

```

In [25]:
```
#  2) Analysing Attrition Column ===>>
```

In [26]:
```
df['attrition'].unique()
# here we can see the number of unique values present in the column are only two [yes] [no]
```

Out[26]: array(['Yes', 'No'], dtype=object)

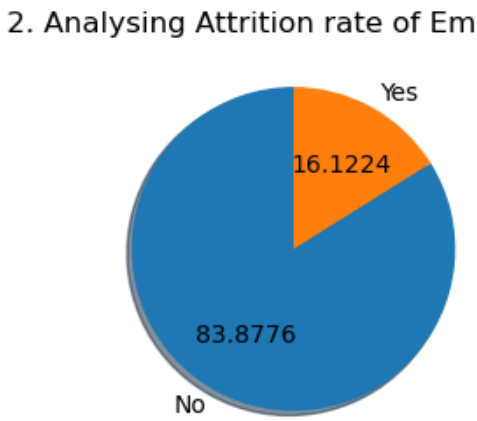
In [27]:
```
df['attrition'].nunique()
#  No. of unique values present in the column is 2
```

Out[27]: 2

In [28]:
```
df['attrition'].value_counts()
```

Out[28]:
```
No     1233
Yes     237
Name: attrition, dtype: int64
```

In [29]:
```
plt.figure(figsize=(3,3))
plt.title('\n 2. Analysing Attrition rate of Employees')
plt.pie(df['attrition'].value_counts(),startangle=90,autopct='%.4f',labels=['No', 'Yes'],shadow
plt.show()

# here as we know the high attrition is problematic for the companies.
# but in this case the YES attrition rate is very less = 16.12 %
# where as the ration of NO attrition is verymuch Higher = 83.87 %
# from this we can conclude that the most of the employees are satisfied from their job in thi
#  only few of the emplyees are not satisfied and moved=on to another companie, so we have to
```



In [ ]:
```

```

In [30]:
```
# 3) Analysing Business level ====>>>
```

In [31]:
```
df['business travel'].unique()
# here following we can find the unique values in the 'business travel' column.
```

Out[31]:
```
array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)
```

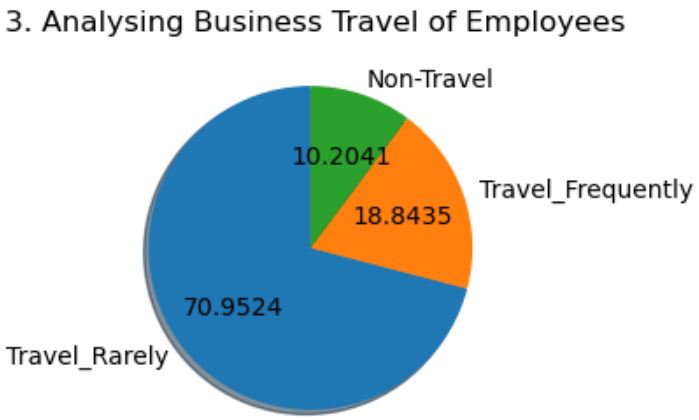
In [32]:
```
df['business travel'].nunique()
#  there are three unique values are present in the column.
```

Out[32]:
```
3
```

In [33]:
```
df['business travel'].value_counts()
# here we can get the no. vlaues for the unique features in the column.
```

Out[33]:
```
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: business travel, dtype: int64
```

In [34]:
```
plt.figure(figsize=(3,3))
plt.title('\n 3. Analysing Business Travel of Employees')
plt.pie(df['business travel'].value_counts(),startangle=90,autopct='%.4f',labels=['Travel_Rare
plt.show()

# here below we can find the frequency of employees for business travel.
# & we can find that out of the total employess, therer are :
#  1- Employees who Travel's Rarely are = 70.95 %
#  2- Employess who Travels's Frequently are = 18.84 %
#  3- Employees who are Non-business Traveler are = 10.20 %

#  that mean Most of the employees are Rarely Travel for business (they may be having sitting
```



In [ ]:
```
```

In [35]:
```
# Analysing Daily Rate =====>>>
```

In [36]:
```
df['daily rate'].nunique()
# out of 1470 , there are 886 unique values are present in the column name.
```

Out[36]: 886

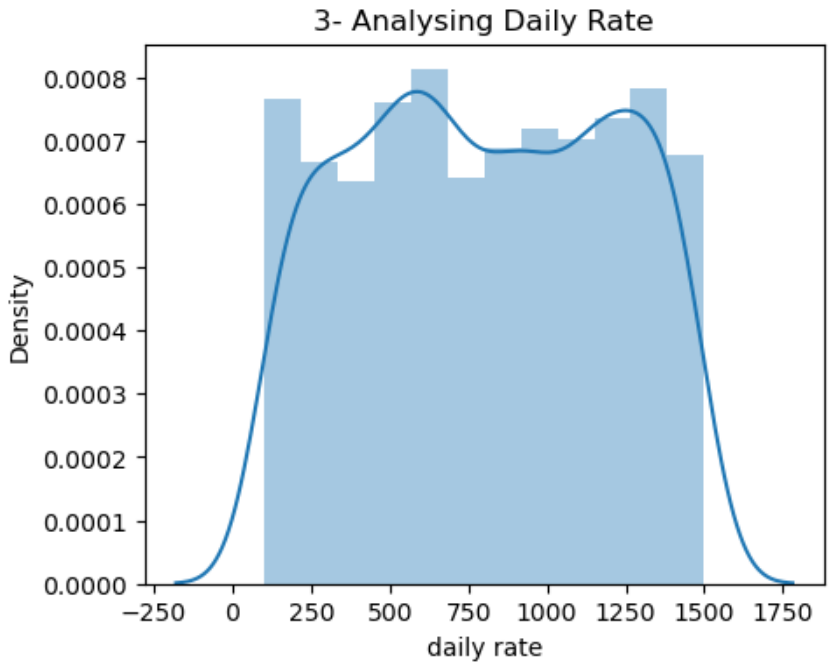
In [37]:
```
df['daily rate'].min()

# here 102 is the minimum daily rate present in the column
```

Out[37]: 102

In [38]:
```
df['daily rate'].max()
#  and maximum daily rate is 1499
```

Out[38]: 1499

In [39]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('3- Analysing Daily Rate')
sns.distplot(df['daily rate'])
# plt.xlabel('Average Price', fontsize = 10)
# plt.xticks(rotation=0,ha ='center')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can see the density of daily rate is almost e
```



In [ ]:
```

```

In [40]:
```
#  4) Analysing Department column ====>>>
```

In [41]:
```
df['department'].unique()
# here following we can get the name of all unique departments present inside the column.
```

Out[41]:
```
array(['Sales', 'Research & Development', 'Human Resources'], dtype=object)
```

In [42]:
```
df['department'].nunique()
# and there are three unique departments presents in the dataset
```

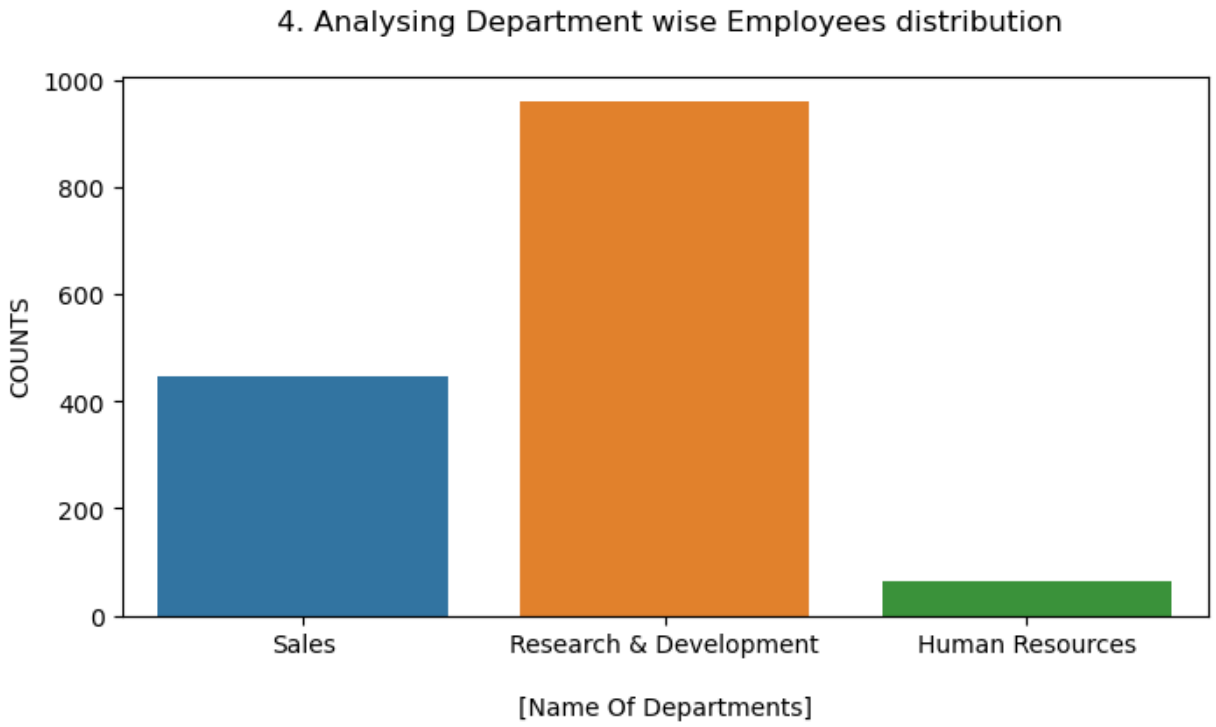
Out[42]:
```
3
```

In [43]:
```
df['department'].value_counts()
# here we can see the no. of employees working the following departments
```

Out[43]:
```
Research & Development    961
Sales                     446
Human Resources            63
Name: department, dtype: int64
```

In [44]:
```
plt.figure (figsize = (8,4), facecolor = "white")
plt.title('\n 4. Analysing Department wise Employees distribution\n')
sns.countplot(x='department', data = df)
plt.xlabel('\n[Name Of Departments]', fontsize=10)
# plt.xticks(rotation=30, ha = 'right')
plt.ylabel('COUNTS',fontsize=10)
# plt.yticks(rotation=30, ha = 'right')

# Here we can see that the most/maximum no. of employees are working in RESEARCH & DEVELOPEMENT
# Then on second highest in SALES TEAM = 446 employees
# then on 3rd level HUMAN RESOURCES = 63 only

# due to which we can CONCLUDE that company's EXPENDITURE most highest on RESEARCH & DEVELOPEMENT
#                                                  (which means company may focus more on RESEARCH
# Secondly , company hired 446 employees for his SALES DEPARTMENT
# Then HUMAN RESOURCE DEPARTMENT which is = 63 employees.
```

Out[44]:
```
Text(0, 0.5, 'COUNTS')
```



In [ ]:
```
```

In [45]:
```
#  5) Analysing Office Distance from home ====>>
```

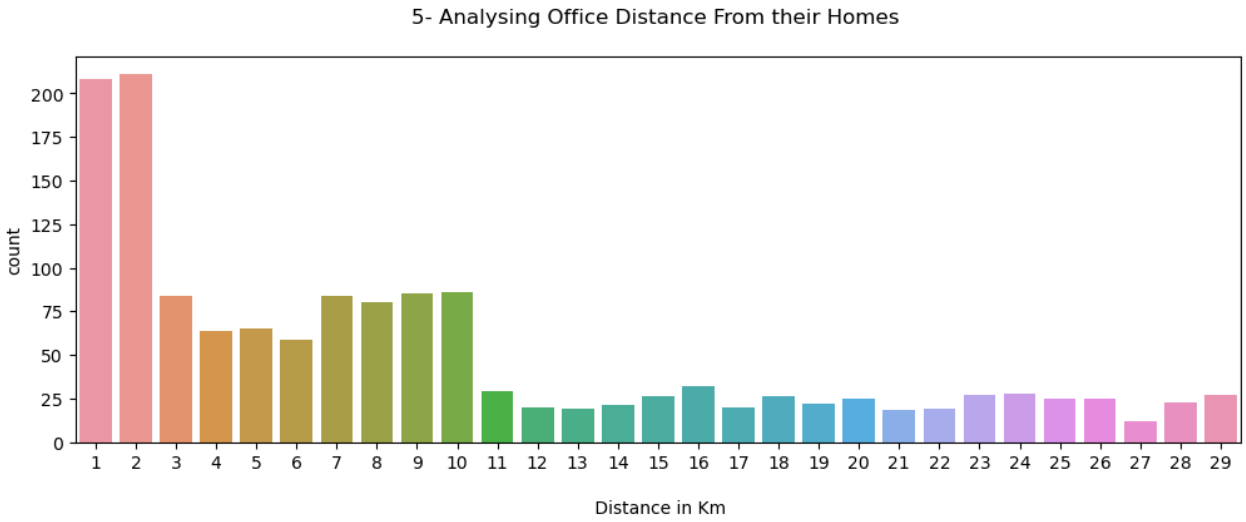
In [46]:
```
df['distance from home'].nunique()
# Ooh its an intresting fact that out of 1470 only 29 unique values are present for 'office dis
```

Out[46]:
```
29
```

In [47]:
```
df['distance from home'].value_counts()
```

Out[47]:
```
2     211
1     208
10     86
9      85
3      84
7      84
8      80
5      65
4      64
6      59
16     32
11     29
24     28
23     27
29     27
15     26
18     26
26     25
25     25
20     25
28     23
19     22
14     21
12     20
17     20
22     19
13     19
21     18
27     12
Name: distance from home, dtype: int64
```

In [48]:
```
plt.figure(figsize = (12,4), facecolor='white')
plt.title('\n5- Analysing Office Distance From their Homes \n')
sns.countplot(x='distance from home', data=df)
plt.xlabel('\n Distance in Km', fontsize = 10)
# plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below Countplot we can find that ,
#  The Highest Number  of Employees are Living in around = 1 -to- 2 Km. from office.
#  The Maximum no. of Employees are living around = 1 -to- 10 km from office.
#  The Distance of home of few of the employees is around = 29 km from office.
```



In [ ]:
```
```

In [49]:
```
#  6) Analysing Educational Qualification of Employees =====>>
```

In [50]:
```
df['education'].unique()
# there are five unique values are present in the mentioned column
# it may be coded therfore we are getting values in numbers.
```

Out[50]:
```
array([2, 1, 4, 3, 5], dtype=int64)
```

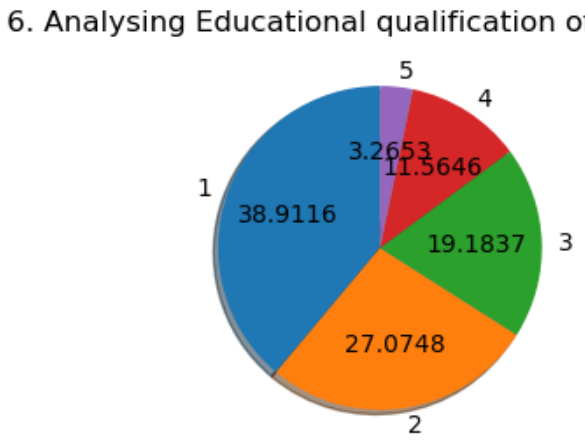
In [51]:
```
df['education'].nunique()
```

Out[51]:
```
5
```

In [52]:
```
df['education'].value_counts()
# here we can see that the highest number of employees qualification is with code'3'>'4'>'2'>'1
# there are only 48 employees whose educational qualification is with code '5'
```

Out[52]:
```
3    572
4    398
2    282
1    170
5     48
Name: education, dtype: int64
```

In [53]:
```
plt.figure(figsize=(3,3))
plt.title('\n 6. Analysing Educational qualification of Employees')
plt.pie(df['education'].value_counts(),startangle=90,autopct='%.4f',labels=['1', '2','3','4','
plt.show()

# here below we can find the frequency of employees for business travel.
# & we can find that out of the total employess, therer are :
#  1- Employees who's educational qualification with code '1 ' is = 38.91 %
#  2- Employees who's educational qualification with code '2 ' is = 27 %
#  3- Employees who's educational qualification with code '3 ' is = 19.18 %
#  4- Employees who's educational qualification with code '4 ' is = 11.56 %
#  5- Employees who's educational qualification with code '5 ' is = 3.26 %
#  that mean Most of the employees are qualified with code '1' & '2'
```



In [ ]:
```
```

In [54]:
```
# 7) Analysing Education Field Of Employee =====>>
```

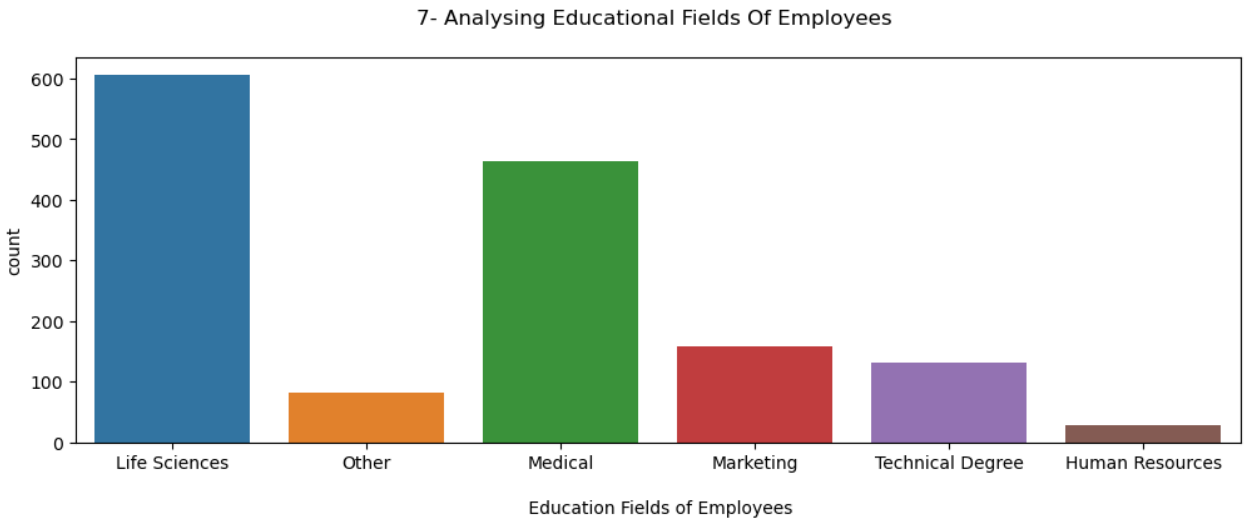
In [55]:
```
df['education field'].unique()
# here below we can find that there are 6 unique educational field are present in the column.
```

Out[55]:
```
array(['Life Sciences', 'Other', 'Medical', 'Marketing',
       'Technical Degree', 'Human Resources'], dtype=object)
```

In [56]:
```
df['education field'].value_counts()
#  here from the below we can find most of the employees are from LIFE SCIENCE & MEDICAL
```

Out[56]:
```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: education field, dtype: int64
```

In [57]:
```
plt.figure(figsize = (12,4), facecolor='white')
plt.title('\n7- Analysing Educational Fields Of Employees \n')
sns.countplot(x='education field', data=df)
plt.xlabel('\n Education Fields of Employees', fontsize = 10)
# plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below Countplot we can find that ,
#  The Highest Number  of Employees are from LIFE SCIENCE & MEDICAL = 606 & 464.
#  Then from MARKETTING & TECHNIQUAL & OTHER fields.
# from this we can conlcude 2 things that :
#         1) Most of the employees of RESEARCH & DEVELOPEMENT DEPARTMENT are from LIFESCIENCE &
#         2) The company could be a PHARMACEAUTICAL / RESEARCH COMPANY.
```



In [ ]:
```

```

In [58]:
```
# 8) Analysing Employee Count ====>>>
```

In [59]:
```
df['employeecount'].unique()
```

Out[59]:
```
array([1], dtype=int64)
```

In [60]:
```
df['employeecount'].nunique()
# there is only 1 unique va;ue present in this column.
```

Out[60]:
```
1
```

In [61]:
```
df['employeecount'].value_counts()
```

Out[61]:
```
1    1470
Name: employeecount, dtype: int64
```

In [62]:
```
# it could be not a relevant column, we can chek it further & if it is not important then we c
```

In [63]:
```
#  9) Analysing employee number ====>>>>
```

In [64]:
```
df['employee number'].unique()
```

Out[64]:
```
array([   1,    2,    4, ..., 2064, 2065, 2068], dtype=int64)
```

In [65]:
```
df['employee number'].nunique()
# i found this column is nothing but seqential no. of employees, i think this not a releven co
```

Out[65]:
```
1470
```

In [ ]:
```

```

In [66]:
```
# 10) Analysing Enviornment Satisfaction =====>>>
```

In [67]:
```
df['environment satisfaction']
```

Out[67]:
```
0       2
1       3
2       4
3       4
4       1
       ..
1465    3
1466    4
1467    2
1468    4
1469    2
Name: environment satisfaction, Length: 1470, dtype: int64
```

In [68]:
```
df['environment satisfaction'].unique()
# there are 4 unique values are present in the columnn.
```

Out[68]:
```
array([2, 3, 4, 1], dtype=int64)
```

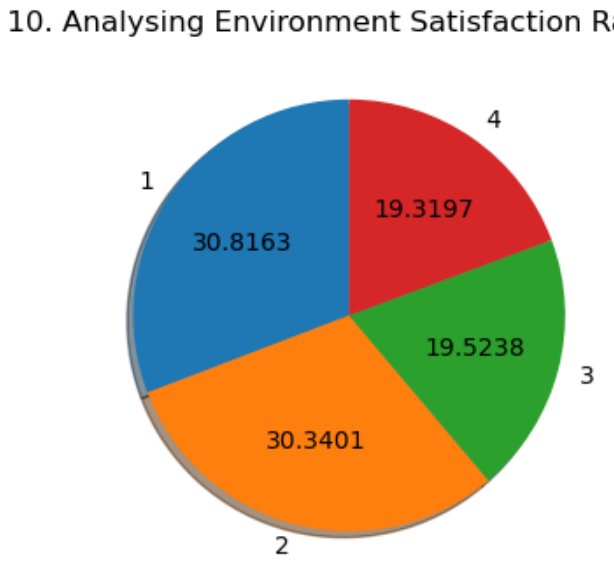
In [69]:
```
df['environment satisfaction'].nunique()
# no. of unique values are 4
```

Out[69]:
```
4
```

In [70]:
```
df['environment satisfaction'].value_counts()
# here we can find that 'Environment Satisfaction' is a categorical column, where 'Satsfaction
#  into ratings given by employees is=== 1,2,3,4....
```

Out[70]:
```
3    453
4    446
2    287
1    284
Name: environment satisfaction, dtype: int64
```

In [71]:
```
plt.figure(figsize=(4,4))
plt.title('\n 10. Analysing Environment Satisfaction Ratings')
plt.pie(df['environment satisfaction'].value_counts(),startangle=90,autopct='%.4f', labels=['1
plt.show()

# here below we can find the % distribution for ENVIRONMENT SATISFACTION.
#  1) The Highest Rating-(1) given for Enviornment Satisfaction from % of employees is = 30.81
# 2) The Second Rating-(2) given for Enviornment Satisfaction from % of employees is = 30.34 %
#  3) then third Rating-(3) given for Enviornment Satisfaction from % of employees is = 19.52 %
#  4) fourth Rating-(4) given for Enviornment Satisfaction from % of employees is = 19.31 %

# The Maximum percentage for ENVIRONMENT SARISAFACTION with Rating (1) is 30.81
#  Minimum is= 19.31 %
```



In [ ]:
```
```

In [72]:
```
# 11) Analysing Gender Column ==========>>>
```

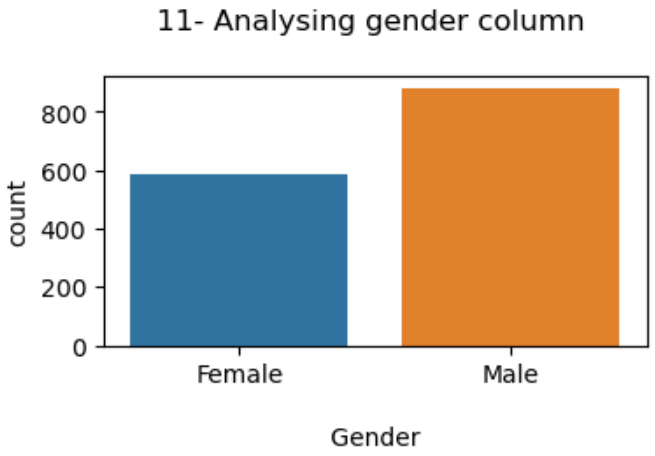
In [73]:
```
df['gender'].unique()
```

Out[73]:
```
array(['Female', 'Male'], dtype=object)
```

In [74]:
```
df['gender'].value_counts()
#  out of Total 1470 employees , there are 882 (male) & 588 (female)
```

Out[74]:
```
Male      882
Female    588
Name: gender, dtype: int64
```

In [75]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n11- Analysing gender column \n')
sns.countplot(x='gender', data=df)
plt.xlabel('\n Gender ', fontsize = 10)
# plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here we can clearly see that , the MALE employees are more as compared to FEMALE employees
```



In [ ]:
```
```

In [76]:
```
# 12) Analysing Hourly Rate =======>>>>
```

In [77]:
```
df['hourly rate']
```

Out[77]:
```
0       94
1       61
2       92
3       56
4       40
        ..
1465    41
1466    42
1467    87
1468    63
1469    82
Name: hourly rate, Length: 1470, dtype: int64
```

In [78]:
```
df['hourly rate'].unique()
```

Out[78]:
```
array([ 94,  61,  92,  56,  40,  79,  81,  67,  44,  84,  49,  31,  93,
        50,  51,  80,  96,  78,  45,  82,  53,  83,  58,  72,  48,  42,
        41,  86,  97,  75,  33,  37,  73,  98,  36,  47,  71,  30,  43,
        99,  59,  95,  57,  76,  87,  66,  55,  32,  52,  70,  62,  64,
        63,  60, 100,  46,  39,  77,  35,  91,  54,  34,  90,  65,  88,
        85,  89,  68,  69,  74,  38], dtype=int64)
```

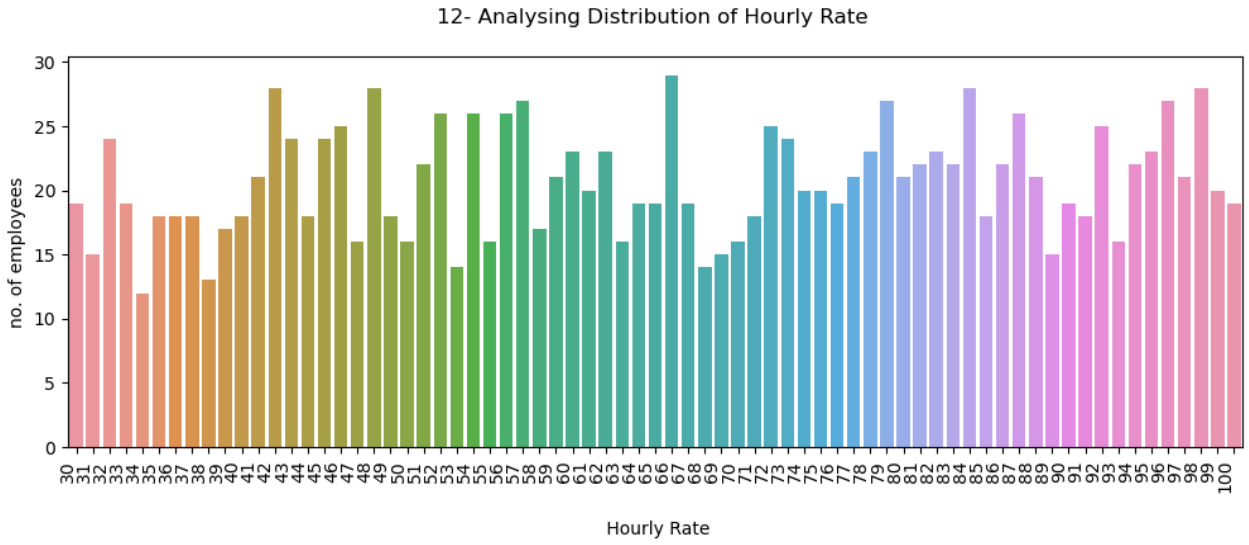
In [79]:
```
df['hourly rate'].nunique()
#  Outof Total 1470 values , there are 71 unique values are present in the column
```

Out[79]:
```
71
```

In [80]:
```
df['hourly rate'].value_counts()
```

Out[80]:
```
66    29
98    28
42    28
48    28
84    28
      ..
31    15
53    14
68    14
38    13
34    12
Name: hourly rate, Length: 71, dtype: int64
```

In [81]:
```
plt.figure(figsize = (12,4), facecolor='white')
plt.title('\n12- Analysing Distribution of Hourly Rate \n')
sns.countplot(x='hourly rate', data=df)
plt.xlabel('\n Hourly Rate', fontsize = 10)
plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below Countplot we can find that , the distribution of 'hourly rate' v/s 'no.
```



In [ ]:
```
```

In [82]:
```
# 13) Analysing Job Enviornment =====>>>
```

In [83]:
```
df['job involvement']
```

Out[83]:
```
0       3
1       2
2       2
3       3
4       3
       ..
1465    4
1466    2
1467    4
1468    2
1469    4
Name: job involvement, Length: 1470, dtype: int64
```

In [84]:
```
df['job involvement'].unique()
# here we can find categorical column, with '1', '2', '3', '4' values
# i think it a rating or codes assingned with 'job involvement' of employee
```

Out[84]:
```
array([3, 2, 4, 1], dtype=int64)
```

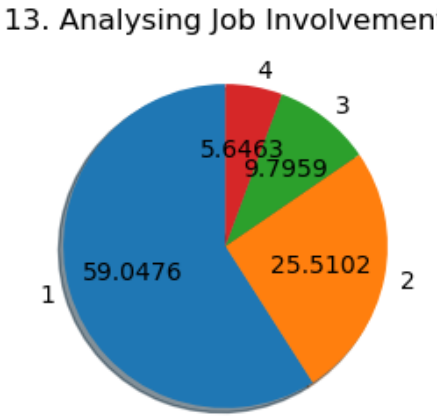
In [85]:
```
df['job involvement'].nunique()
# there are four no. of unique categories are present in the dataset.
```

Out[85]:
```
4
```

In [86]:
```
df['job involvement'].value_counts()
# highesht no. of employees are assigned with code'3'
```

Out[86]:
```
3    868
2    375
4    144
1     83
Name: job involvement, dtype: int64
```

In [87]:
```
plt.figure(figsize=(3,3))
plt.title('13. Analysing Job Involvement')
plt.pie(df['job involvement'].value_counts(),startangle=90,autopct='%.4f',labels=['1','2','3',
plt.show()

# here we can clearly see that highest (59.04%) employees are assigned with 'job involvement c
```



In [ ]:
```

```

In [88]:
```
# 14) Analysing Job Level ====>>>
```

In [89]:
```
df['job level']
```

Out[89]:
```
0       2
1       2
2       1
3       1
4       1
       ..
1465    2
1466    3
1467    2
1468    2
1469    2
Name: job level, Length: 1470, dtype: int64
```

In [90]:
```
df['job level'].unique()
#  it also a categorical column , with some job level defining categories
```

Out[90]:
```
array([2, 1, 3, 4, 5], dtype=int64)
```

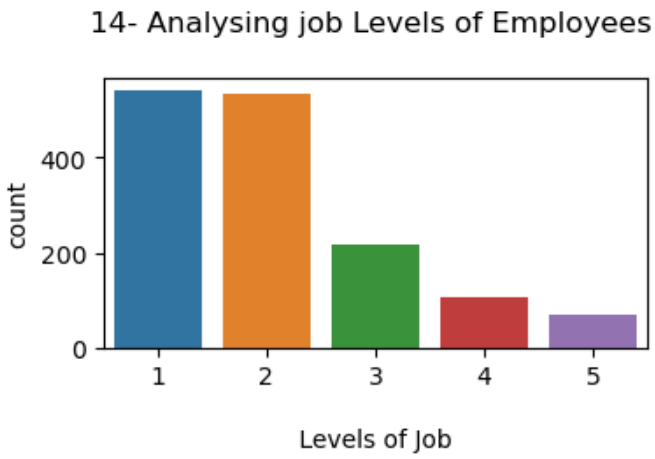
In [91]:
```
df['job level'].nunique()
# there are five number of unique vlaues are present in the column for defining 'job level'
```

Out[91]:
```
5
```

In [92]:
```
df['job level'].value_counts()
# most of the employees are having 1 & 2 level of job
```

Out[92]:
```
1    543
2    534
3    218
4    106
5     69
Name: job level, dtype: int64
```

In [93]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n14- Analysing job Levels of Employees \n')
sns.countplot(x='job level', data=df)
plt.xlabel('\n Levels of Job ', fontsize = 10)
# plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here we can clearly see that , the most of the employees are in 1 & 2 level of job.
#  it could be RESEARCH & DEVELOPEMENT category, because earlier we seen that most of the emplo
```



In [ ]:
```
```

In [94]:
```
#  15) Analysing Job Role =====>>
```

In [95]:
```
df['job role'].unique()
# there are several 9 different roles of job , present inside the column.
```

Out[95]:
```
array(['Sales Executive', 'Research Scientist', 'Laboratory Technician',
       'Manufacturing Director', 'Healthcare Representative', 'Manager',
       'Sales Representative', 'Research Director', 'Human Resources'],
      dtype=object)
```

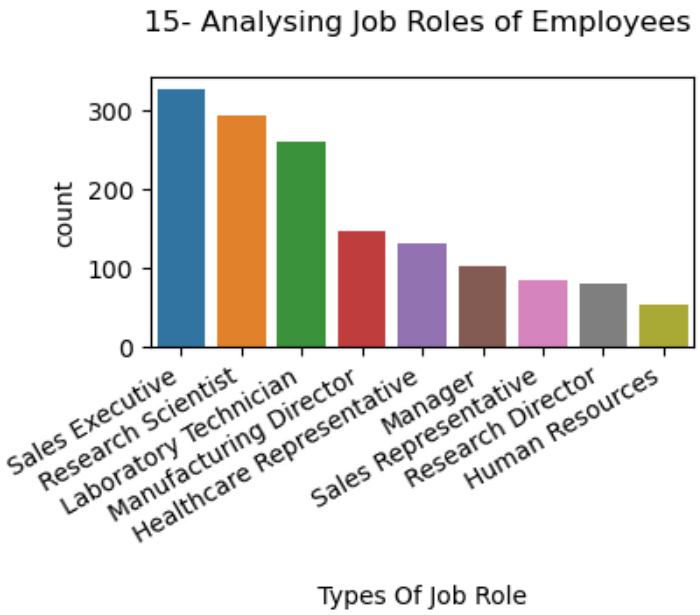
In [96]:
```
df['job role'].nunique()
# 9 no. of unique job roles are present inside the column.
```

Out[96]:
```
9
```

In [97]:
```
df['job role'].value_counts()
# here we can find the no. of employees in different job roles.
```

Out[97]:
```
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: job role, dtype: int64
```

In [98]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n15- Analysing Job Roles of Employees \n')
sns.countplot(x='job role', data=df)
plt.xlabel('\n Types Of Job Role ', fontsize = 10)
plt.xticks(rotation=30,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here we can clearly see that , the most of the employees are in the role of 'sales executive
# ...'Laboratory Technishian'....then it will go further decreasing to other roles.
```



In [ ]:
```
```

In [99]:
```
#  16) Analysing Job Satisfaction =====>>>
```

In [100]:
```
df['job satisfaction']
```

Out[100]:
```
0       4
1       2
2       3
3       3
4       2
       ..
1465    4
1466    1
1467    2
1468    2
1469    3
Name: job satisfaction, Length: 1470, dtype: int64
```

In [101]:
```
df['job satisfaction'].unique()
# this is also an categorical column with 1,2,3 & 4 ratings
```

Out[101]:
```
array([4, 2, 3, 1], dtype=int64)
```

In [102]:
```
df['job satisfaction'].nunique()
```

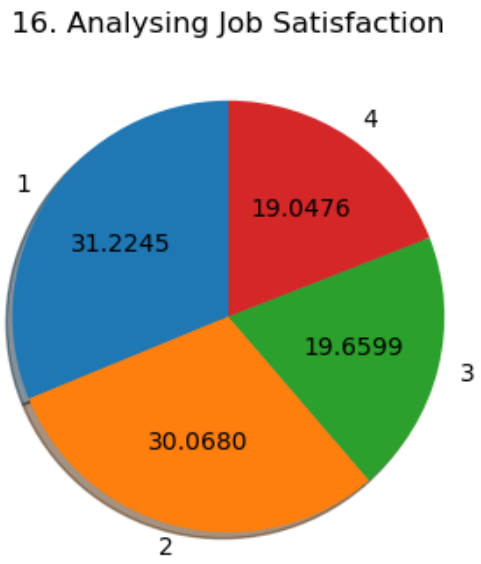
Out[102]: 4

In [103]:
```
df['job satisfaction'].value_counts()
```

Out[103]:
```
4    459
3    442
1    289
2    280
Name: job satisfaction, dtype: int64
```

In [ ]:

In [104]:
```
plt.figure(figsize=(4,4))
plt.title('16. Analysing Job Satisfaction')
plt.pie(df['job satisfaction'].value_counts(),startangle=90,autopct='%.4f',labels=['1','2','3'
plt.show()

# here we can find the percentage with different levels of job satisfaction:
# No. of percentage of employees with job satisfaction code(1) is = 31.22%
# No. of percentage of employees with job satisfaction code(2) is = 30.06%
# No. of percentage of employees with job satisfaction code(3) is = 19.65%
# No. of percentage of employees with job satisfaction code(4) is = 19.04%
```



In [ ]:

In [105]:
```
# 17) Analysing Marital Status =======>>>>>
```

In [106]:
```
df['marital status']
```

Out[106]:
```
0        Single
1       Married
2        Single
3       Married
4       Married
         ...
1465    Married
1466    Married
1467    Married
1468    Married
1469    Married
Name: marital status, Length: 1470, dtype: object
```

In [107]:
```
df['marital status'].unique()
# there are 3 categories inside this column.
```

Out[107]:
```
array(['Single', 'Married', 'Divorced'], dtype=object)
```

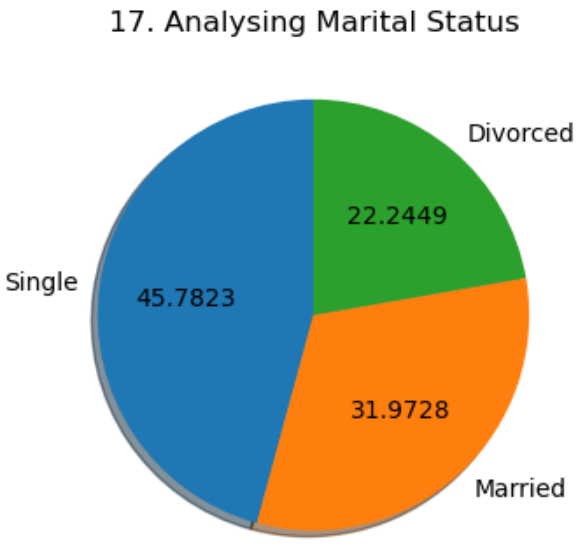
In [108]:
```
df['marital status'].nunique()
```

Out[108]:
```
3
```

In [109]:
```
df['marital status'].value_counts()
# here we can find the no. employee counts for the 3 difeerent categories.
```

Out[109]:
```
Married     673
Single      470
Divorced    327
Name: marital status, dtype: int64
```

In [110]:
```
plt.figure(figsize=(4,4))
plt.title('17. Analysing Marital Status')
plt.pie(df['marital status'].value_counts(),startangle=90,autopct='%.4f',labels=['Single','Mar
plt.show()

# here we can find the percentage with different Marital Status:
# there are most of the employees are SINGLE = 45.78%
# then Married employees are = 31.97%
#  and Divorced Employees are = 22.24 %
```



In [ ]:
```
```

In [111]:
```
# 18) Analysing Monthly Income of Employees ====>>>
```

In [112]:
```
df['monthly income'].unique()
```

Out[112]:
```
array([5993, 5130, 2090, ..., 9991, 5390, 4404], dtype=int64)
```

In [113]:
```
df['monthly income'].nunique()
# out of 1740 there are 1349 unique vlaues are present.
# this column can't be consider as a categorical column.
```

Out[113]:
```
1349
```

In [114]:
```
df['monthly income'].min()
# the minimum monthly income of any employee is $ 1009
```

Out[114]:
```
1009
```

In [115]:
```
df['monthly income'].max()
# and the maximum monthly income of any employee is $ 19,999
```

Out[115]:
```
19999
```

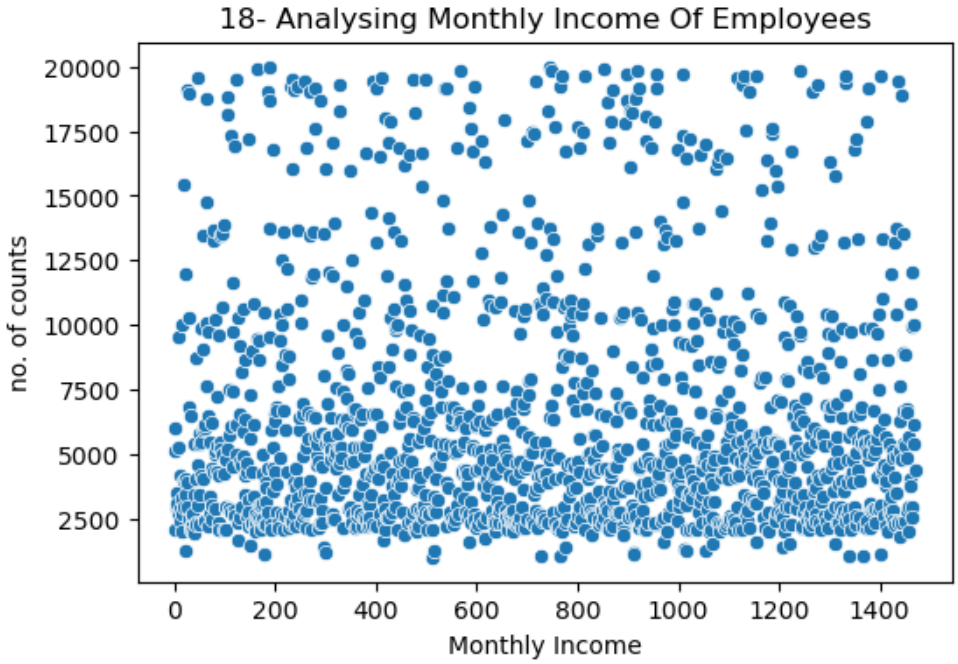
In [116]:
```
df['monthly income'].median()
#  here we can find the MEDIAN of Monthly Income of all Employees is 4,919 $
```

Out[116]:
```
4919.0
```

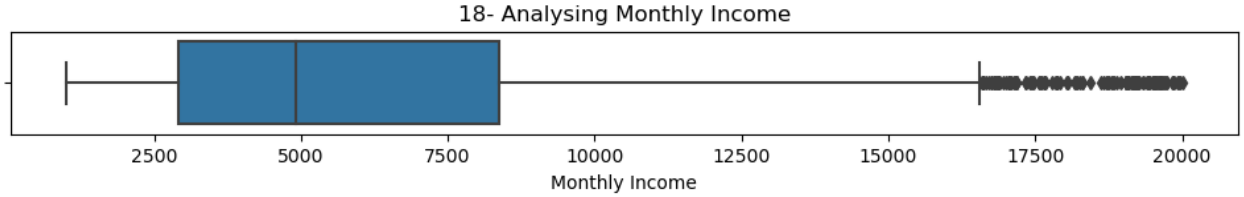
In [117]:
```
df['monthly income'].mean()
# the MEAN of the Monthly Iincome of Employees is $ 6,500 approx
```

Out[117]:
```
6502.931292517007
```

In [118]:
```
plt.figure(figsize = (6,4), facecolor='white')
plt.title('18- Analysing Monthly Income Of Employees')
sns.scatterplot(df.index,df['monthly income'])
plt.xlabel('Monthly Income', fontsize = 10)
# plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here in the below scatterplot it is difficult to identify.
```



In [119]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('18- Analysing Monthly Income')
sns.boxplot(x='monthly income', data=df)
plt.xlabel('Monthly Income', fontsize = 10)
# plt.xticks(rotation=90,ha ='center')
# plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below boxplot we can find that :
#  minimum monthly income is below $ 2500 = $ 1009
#  Highest Monthly income is near to = $ 2000
#  25th percentile of montly income is = near about $2500
# 50th percentile (Median) of monthly income is = near about $ 5000
# 75th percentile of monthly income is = near about $ 7800 (i.e 75 percentile employees are ge
```



In [ ]:

In [120]:
```
#  19) Analysing Monthly Rate =====>>>>
```

In [121]:
```
df['monthly rate']
```

Out[121]:
```
0       19479
1       24907
2        2396
3       23159
4       16632
        ...
1465    12290
1466    21457
1467     5174
1468    13243
1469    10228
Name: monthly rate, Length: 1470, dtype: int64
```

In [122]:
```
df['monthly rate'].unique()
```

Out[122]:
```
array([19479, 24907,  2396, ...,  5174, 13243, 10228], dtype=int64)
```

In [123]:
```
df['monthly rate'].nunique()
# out of 1470 values there are 1427 unique vlaues present in the column. that means it is not c
```

Out[123]:
```
1427
```

In [124]:
```
df['monthly rate'].min()
#  the 'minimum monthyl rate value' present in the column is 2094.
```

Out[124]:
```
2094
```

In [125]:
```
df['monthly rate'].max()
# and the maximum monthly rate assigned for any employee is 26,999
```

Out[125]:
```
26999
```

In [126]:
```
df['monthly rate'].median()
# the median for monthly rate is 14,235
```

Out[126]:
```
14235.5
```

In [127]:
```
df['monthly rate'].nlargest(10)
# here we can find below the TOP 10 LARGEST MONTHLYL RATE VALUES.
```

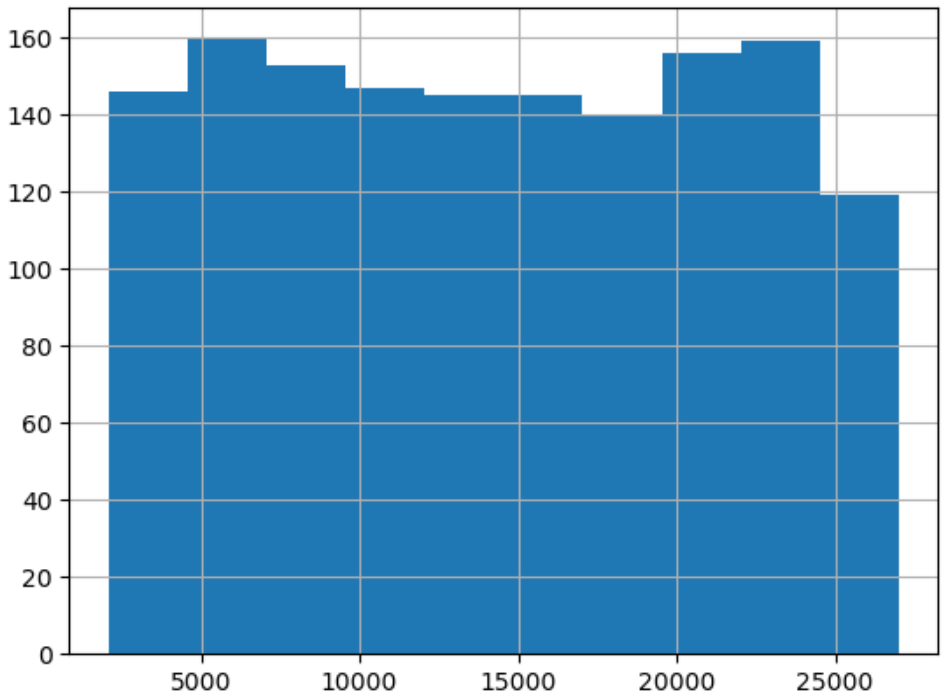
Out[127]:
```
513     26999
1350    26997
808     26968
107     26959
1438    26956
859     26933
386     26914
108     26897
493     26894
1405    26862
Name: monthly rate, dtype: int64
```

In [128]:
```
df['monthly rate'].nsmallest(10)
# and here below we can find the last 10 smallest monthly rate .
```

Out[128]:
```
28      2094
1245    2097
543     2104
459     2112
1381    2122
473     2125
1370    2125
178     2137
396     2227
613     2243
Name: monthly rate, dtype: int64
```

In [129]:
```
df['monthly rate'].hist()
# here below in the histogram we can find that the highest density is lying in between 5000-25
```

Out[129]: <AxesSubplot:>



In [130]:
```
df['monthly rate'].empty
# there are no empty spaces inside the column.
```

Out[130]: False

In [131]:
```
# 20) Analysing Number of Companies Worked =======>>
```

In [132]:
```
df['num companies worked']
```

Out[132]:
```
0       8
1       1
2       6
3       1
4       9
       ..
1465    4
1466    4
1467    1
1468    2
1469    2
Name: num companies worked, Length: 1470, dtype: int64
```

In [133]:
```
df['num companies worked'].unique()
#  following are the unique valaues present inside the column
```

Out[133]:
```
array([8, 1, 6, 9, 0, 4, 5, 2, 7, 3], dtype=int64)
```

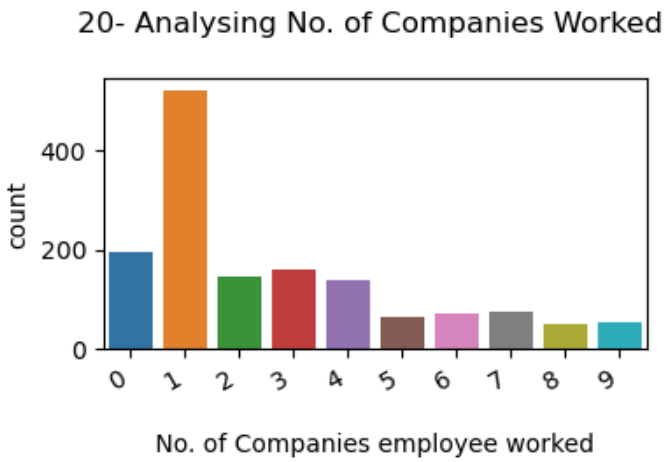
In [134]:
```
df['num companies worked'].nunique()
# there are 10 unique values are present inside the column.
```

Out[134]:
```
10
```

In [135]:
```
df['num companies worked'].value_counts()
# i think this column is belongs to , in how manys companies an employee is worked yet
#  there are 197 freshers are in this company
# the highest number is 521 employees who worked in 1 company before this job.
```

Out[135]:
```
1    521
0    197
3    159
2    146
4    139
7     74
6     70
5     63
9     52
8     49
Name: num companies worked, dtype: int64
```

In [136]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n20- Analysing No. of Companies Worked \n')
sns.countplot(x='num companies worked', data=df)
plt.xlabel('\n No. of Companies employee worked ', fontsize = 10)
plt.xticks(rotation=30,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# i think this column is belongs to , in how manys companies an employee is worked yet
#  there are 197 freshers are in this company
# the highest number is 521 employees who worked in 1 company before this job.
```



In [ ]:

In [ ]:

In [137]:
```
#  21) Analysing Over18 ====>>>
```

In [138]:
```
df['over18']
```

Out[138]:
```
0       Y
1       Y
2       Y
3       Y
4       Y
       ..
1465    Y
1466    Y
1467    Y
1468    Y
1469    Y
Name: over18, Length: 1470, dtype: object
```

In [139]:
```
df['over18'].unique()
```

Out[139]:
```
array(['Y'], dtype=object)
```

In [140]:
```
df['over18'].nunique()
#  here we found only 1 vlaue because, offcourse all employees who are working in thi company
```

Out[140]:
```
1
```

In [ ]:
```
```

In [141]:
```
#  22) Analysing overtime column ====>>>
```

In [142]:
```
df['over time']
```

Out[142]:
```
0       Yes
1        No
2       Yes
3       Yes
4        No
       ...
1465     No
1466     No
1467    Yes
1468     No
1469     No
Name: over time, Length: 1470, dtype: object
```

In [143]:
```
df['over time'].unique()
# following are the unique values present inside the column
```

Out[143]:
```
array(['Yes', 'No'], dtype=object)
```

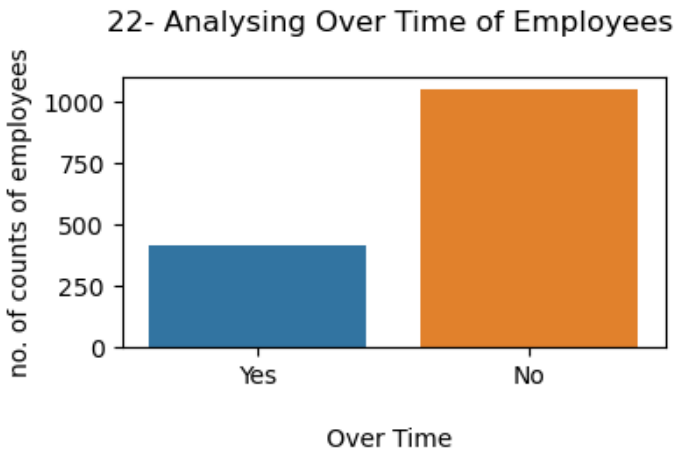
In [144]:
```
df['over time'].nunique()
# there are two unique values are present inside the column
```

Out[144]:
```
2
```

In [145]:
```
df['over time'].value_counts()
# 1054 employees are not overtiming in job
# only 416 employees are over timed
```

Out[145]:
```
No     1054
Yes     416
Name: over time, dtype: int64
```

In [146]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n22- Analysing Over Time of Employees \n')
sns.countplot(x='over time', data=df)
plt.xlabel('\n Over Time  ', fontsize = 10)
# plt.xticks(rotation=30,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here we can fidn that most of the employees are NOT OVERTIMING ...
# only few of them are OVERTIMING
```



In [ ]:
```
```

In [147]:
```
#  23) Analysing Percent Salary Hike =====>>>>
```

In [148]:
```
df['percent salary hike']
```

Out[148]:
```
0       11
1       23
2       15
3       11
4       12
        ..
1465    17
1466    15
1467    20
1468    14
1469    12
Name: percent salary hike, Length: 1470, dtype: int64
```

In [149]:
```
df['percent salary hike'].unique()
```

Out[149]:
```
array([11, 23, 15, 12, 13, 20, 22, 21, 17, 14, 16, 18, 19, 24, 25],
      dtype=int64)
```

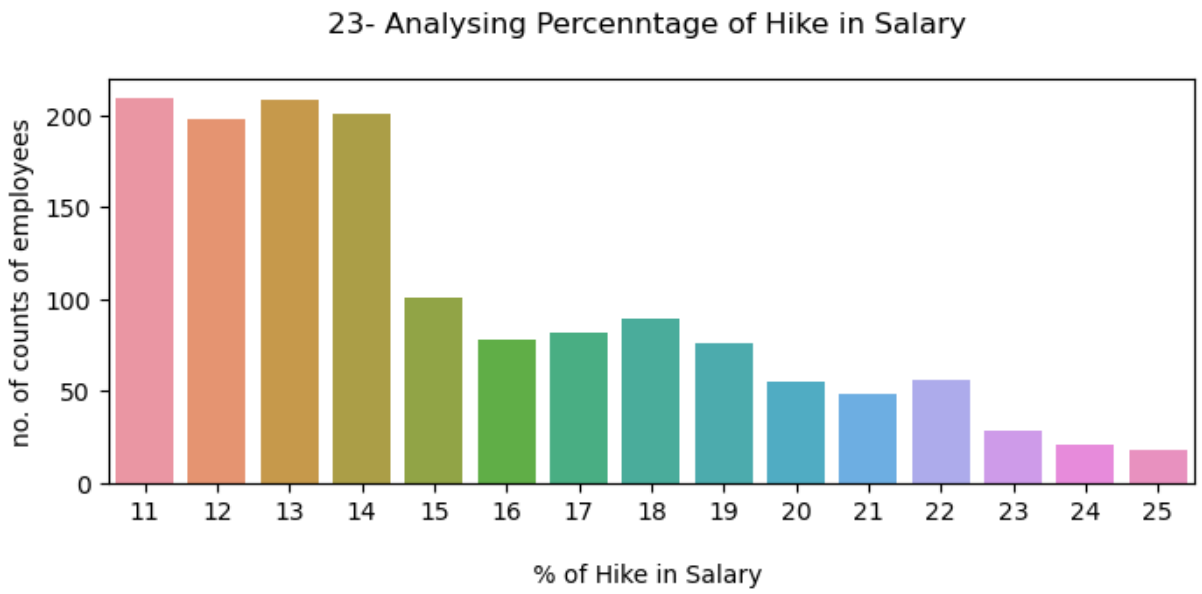
In [150]:
```
df['percent salary hike'].nunique()
```

Out[150]:
```
15
```

In [151]:
```
df['percent salary hike'].value_counts()
# here we can find the no. of employees for 'percent salary hike'
```

Out[151]:
```
11    210
13    209
14    201
12    198
15    101
18     89
17     82
16     78
19     76
22     56
20     55
21     48
23     28
24     21
25     18
Name: percent salary hike, dtype: int64
```

In [152]:
```
plt.figure(figsize = (8,3), facecolor='white')
plt.title('\n23- Analysing Percenntage of Hike in Salary \n')
sns.countplot(x='percent salary hike', data=df)
plt.xlabel('\n % of Hike in Salary  ', fontsize = 10)
# plt.xticks(rotation=30,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here below we can find that most of the employees are getting HIKE OF 11- 14 % of their curre
# only few of the employees are HIKED IN SALARY UPTO 25%
```



In [ ]:
```
```

In [153]:
```
# 24) Analysing Performance Rating Of Employees ======>>>
```

In [154]:
```
df['performance rating']
```

Out[154]:
```
0       3
1       4
2       3
3       3
4       3
       ..
1465    3
1466    3
1467    4
1468    3
1469    3
Name: performance rating, Length: 1470, dtype: int64
```

In [155]:
```
df['performance rating'].unique()
# only 3 & 4 unique values
```

Out[155]:
```
array([3, 4], dtype=int64)
```

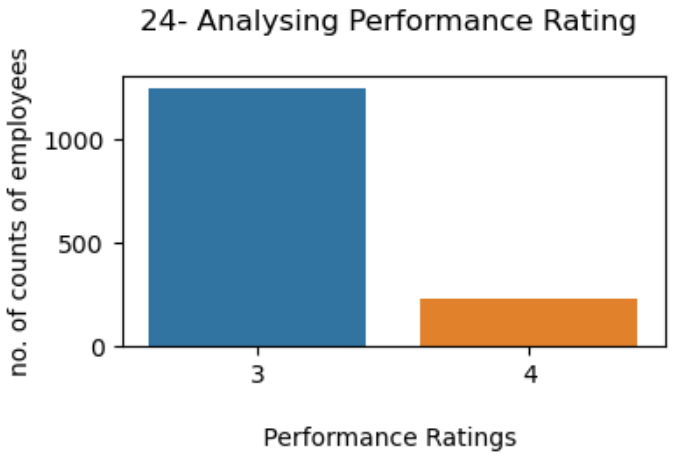
In [156]:
```
df['performance rating'].nunique()
# there are  only 2 no. of unique values are present in the column
```

Out[156]:
```
2
```

In [157]:
```
df['performance rating'].value_counts()
# we can find here that 1244 employees are RATED-3  for their performance
#  & only 226 employees are RATED-4 for their performance
```

Out[157]:
```
3    1244
4     226
Name: performance rating, dtype: int64
```

In [158]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n24- Analysing Performance Rating \n')
sns.countplot(x='performance rating', data=df)
plt.xlabel('\n Performance Ratings  ', fontsize = 10)
# plt.xticks(rotation=30,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below we can find that the Highest no. of employees(1244) are RATE-3 for their performa
# only few of the employees (226) are RATED-4 for their performance.
```



In [ ]:

In [159]:
```
# 25) Analysing Relationship Satisfaction ======>>
```

In [160]:
```
df['relationship satisfaction']
```

Out[160]:
```
0       1
1       4
2       2
3       3
4       4
       ..
1465    3
1466    1
1467    2
1468    4
1469    1
Name: relationship satisfaction, Length: 1470, dtype: int64
```

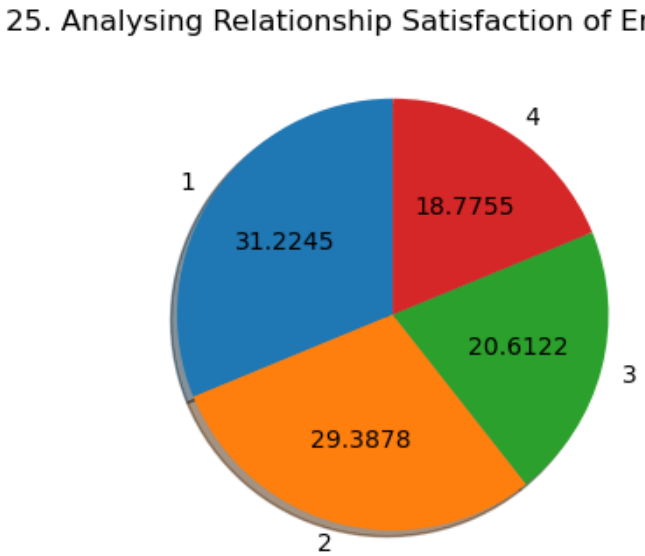
In [161]:
```
df['relationship satisfaction'].unique()
# here in this column the ratings are given to the employees for their RELATIONSHIP BEHAVIOURAL
# so here we can find below 1-to-4 ratings for their relatipnship satisfaction.
```

Out[161]:
```
array([1, 4, 2, 3], dtype=int64)
```

In [162]:
```
df['relationship satisfaction'].nunique()
```

Out[162]:
```
4
```

In [163]:
```
plt.figure(figsize=(4,4))
plt.title('25. Analysing Relationship Satisfaction of Employees')
plt.pie(df['relationship satisfaction'].value_counts(),startangle=90,autopct='%.4f',labels=['1
plt.show()

#  here below in the pie chart we can find that :
# Percentage of employees who RATED-1 for their RELATIONSHIP BEHAVIOURAL SATISFACTION are = 31
# Percentage of employees who RATED-2 for their RELATIONSHIP BEHAVIOURAL SATISFACTION are = 29
# Percentage of employees who RATED-3 for their RELATIONSHIP BEHAVIOURAL SATISFACTION are = 20
# Percentage of employees who RATED-4 for their RELATIONSHIP BEHAVIOURAL SATISFACTION are = 18
```



In [ ]:

In [164]:
```
#  26) Analysing Standard Hours ====>>>
```

In [165]:
```
df['standard hours'].unique()
```

Out[165]:
```
array([80], dtype=int64)
```

In [166]:
```
df['standard hours'].nunique()
# here we can find that the STANDARD WORKING HOURS FOR ANY EMPLOYEE is 80 HOURS IN A MONTH.
# that means if there are 20 working days in a month , then employee can work atlest [4 HOURS
```

Out[166]:
```
1
```

In [ ]:

In [167]:
```
# 27) Analysing Stock Option Level ============>>>>
```

In [168]:
```
df['stock option Level']
```

Out[168]:
```
0       0
1       1
2       0
3       0
4       1
       ..
1465    1
1466    1
1467    1
1468    0
1469    0
Name: stock option Level, Length: 1470, dtype: int64
```

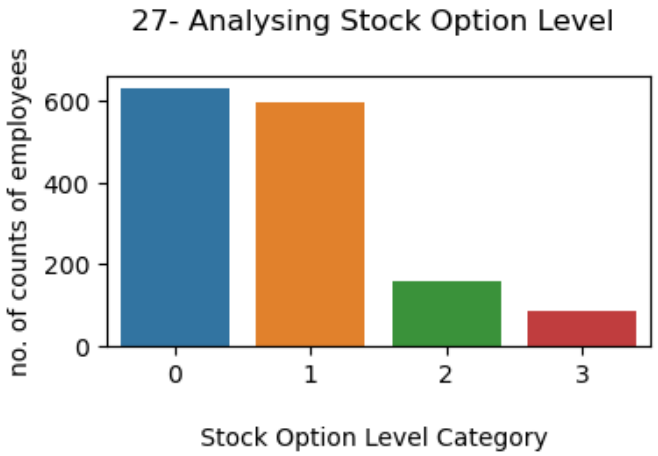
In [169]:
```
df['stock option Level'].unique()
# following unique values are present in the column.
```

Out[169]:
```
array([0, 1, 3, 2], dtype=int64)
```

In [170]:
```
df['stock option Level'].value_counts()
```

Out[170]:
```
0    631
1    596
2    158
3     85
Name: stock option Level, dtype: int64
```

In [171]:
```
plt.figure(figsize = (4,2), facecolor='white')
plt.title('\n27- Analysing Stock Option Level \n')
sns.countplot(x='stock option Level', data=df)
plt.xlabel('\n Stock Option Level Category  ', fontsize = 10)
# plt.xticks(rotation=30,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  Most of the employees are assigned with 0 & 1 Stock Option Level
```



In [ ]:
```

```

In [172]:
```
# 28) Analysing Total Working Year Experiance of employees =======>>>
```

In [173]:
```
df['total working years'].unique()
```

Out[173]:
```
array([ 8, 10,  7,  6, 12,  1, 17,  5,  3, 31, 13,  0, 26, 24, 22,  9, 19,
        2, 23, 14, 15,  4, 29, 28, 21, 25, 20, 11, 16, 37, 38, 30, 40, 18,
       36, 34, 32, 33, 35, 27], dtype=int64)
```

In [174]:
```
df['total working years'].nunique()
# there are 40 no. of unique values for working years are present in the column
```

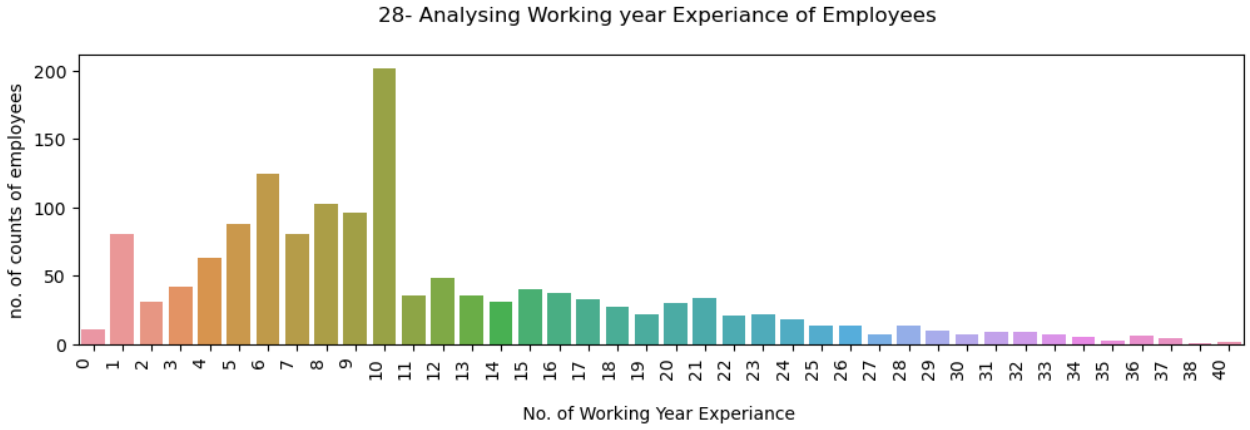
Out[174]:
```
40
```

In [175]:
```
df['total working years'].value_counts()
```

Out[175]:
```
10    202
6     125
8     103
9      96
5      88
7      81
1      81
4      63
12     48
3      42
15     40
16     37
11     36
13     36
21     34
17     33
2      31
14     31
20     30
18     27
19     22
23     22
22     21
24     18
25     14
28     14
26     14
0      11
29     10
31      9
32      9
30      7
33      7
27      7
36      6
34      5
37      4
35      3
40      2
38      1
Name: total working years, dtype: int64
```

In [ ]:
```

```

In [176]:
```
plt.figure(figsize = (12,3), facecolor='white')
plt.title('\n28- Analysing Working year Experiance of Employees \n')
sns.countplot(x='total working years', data=df)
plt.xlabel('\n No. of Working Year Experiance  ', fontsize = 10)
plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below we can find that highest no. of employees are having working experiance of 6-10 y
#  81 no. of emplyees are also with 1 year experiance.
#  there are also few of the employees with the Highest Working Experiance of 40 Years.
```



In [ ]:
```

```

In [177]:
```
#  29) Analysing Employees Training times Last year ======>>>
```

In [178]:
```
df['training times last year'].unique()
# here below we get the unique vlaues of the volumn
```

Out[178]:
```
array([0, 3, 2, 5, 1, 4, 6], dtype=int64)
```

In [179]:
```
df['training times last year'].nunique()
# there are 7 no. of unique values are present in the column
```

Out[179]:
```
7
```

In [180]:
```
df['training times last year'].value_counts()
```

Out[180]:
```
2    547
3    491
4    123
5    119
1     71
6     65
0     54
Name: training times last year, dtype: int64
```

In [181]:
```
plt.figure(figsize = (8,3), facecolor='white')
plt.title('\n29- Analysing Employees Training Times Last Year- 2022 \n')
sns.countplot(x='training times last year', data=df)
plt.xlabel('\n No. of Training Times Of Employees  ', fontsize = 10)
# plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  most of the employees are completed their traoning 2- 3 times last year
```

In [ ]:

In [182]:
```
# 30) Analaysing Work Life Balance ======>>
```

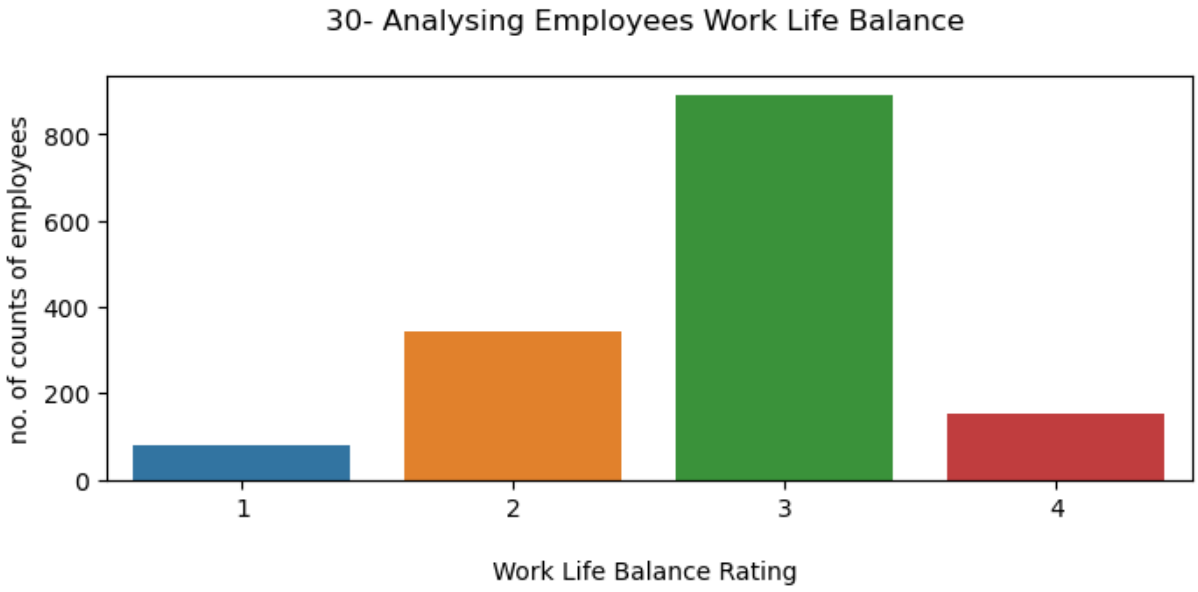
In [183]:
```
df['work life balance'].unique()
```

Out[183]:
```
array([1, 3, 2, 4], dtype=int64)
```

In [184]:
```
df['work life balance'].value_counts()
```

Out[184]:
```
3    893
2    344
4    153
1     80
Name: work life balance, dtype: int64
```

In [185]:
```
plt.figure(figsize = (8,3), facecolor='white')
plt.title('\n30- Analysing Employees Work Life Balance \n')
sns.countplot(x='work life balance', data=df)
plt.xlabel('\n Work Life Balance Rating  ', fontsize = 10)
# plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  Most of the employees (893) are RATED-3 for their WORK LIFE BALANCE
```



In [ ]:
```

```

In [186]:
```
# 31) Analysing Years At Current Company =====>>>>
```

In [187]:
```
df['years at company'].unique()
```

Out[187]:
```
array([ 6, 10,  0,  8,  2,  7,  1,  9,  5,  4, 25,  3, 12, 14, 22, 15, 27,
       21, 17, 11, 13, 37, 16, 20, 40, 24, 33, 19, 36, 18, 29, 31, 32, 34,
       26, 30, 23], dtype=int64)
```

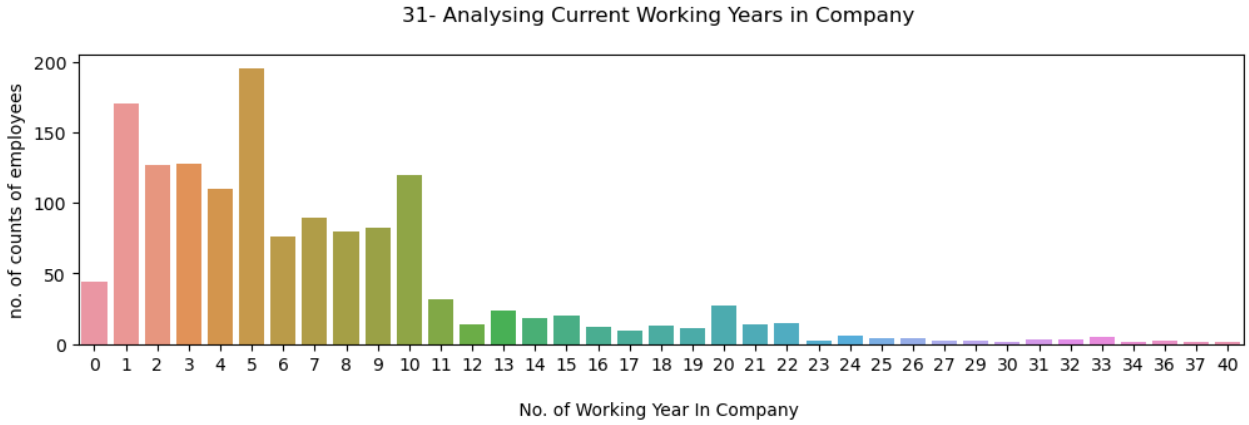
In [188]:
```
df['years at company'].nunique()
```

Out[188]:
```
37
```

In [189]:
```
df['years at company'].value_counts()
```

Out[189]:
```
5     196
1     171
3     128
2     127
10    120
4     110
7      90
9      82
8      80
6      76
0      44
11     32
20     27
13     24
15     20
14     18
22     15
12     14
21     14
18     13
16     12
19     11
17      9
24      6
33      5
25      4
26      4
31      3
32      3
27      2
36      2
29      2
23      2
37      1
40      1
34      1
30      1
Name: years at company, dtype: int64
```

In [190]:
```
plt.figure(figsize = (12,3), facecolor='white')
plt.title('\n31- Analysing Current Working Years in Company \n')
sns.countplot(x='years at company', data=df)
plt.xlabel('\n No. of Working Year In Company  ', fontsize = 10)
# plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  Most of the employees are working in this company from 1- 10 years.
# highest number of employees are working from 1 (171 employees) & 5 (196 employees) year
# few of the employees are also working in this company sinc last 40 years.
```



In [ ]:
```

```

In [191]:
```
#  32) Analysing Years in Current Role ================>>>>>
```

In [192]:
```
df['years in current role'].unique()
```

Out[192]:
```
array([ 4,  7,  0,  2,  5,  9,  8,  3,  6, 13,  1, 15, 14, 16, 11, 10, 12,
       18, 17], dtype=int64)
```

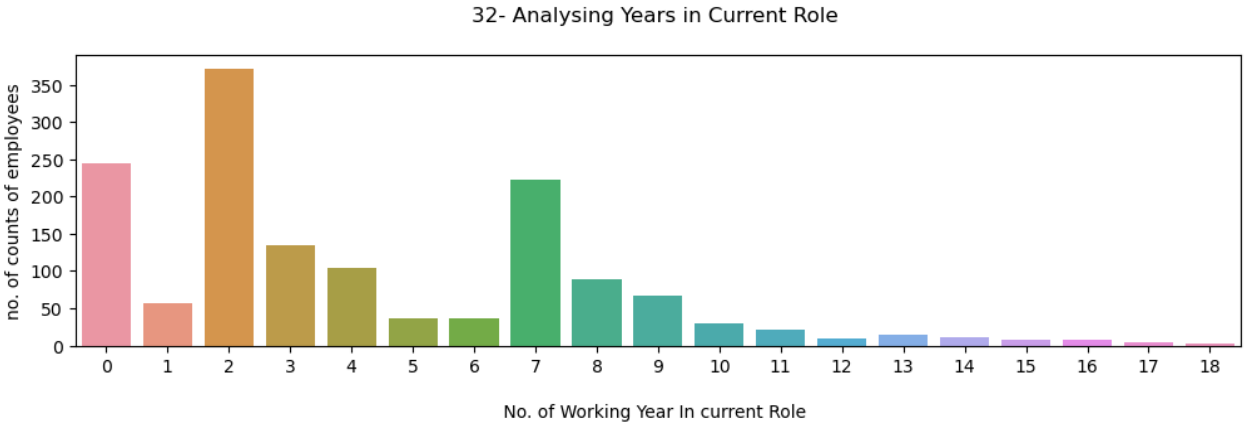
In [193]:
```
df['years in current role'].nunique()
# 19 no. of unique values are present in the current column
```

Out[193]:
```
19
```

In [194]:
```
df['years in current role'].value_counts()
```

Out[194]:
```
2     372
0     244
7     222
3     135
4     104
8      89
9      67
1      57
6      37
5      36
10     29
11     22
13     14
14     11
12     10
15      8
16      7
17      4
18      2
Name: years in current role, dtype: int64
```

In [195]:
```
plt.figure(figsize = (12,3), facecolor='white')
plt.title('\n32- Analysing Years in Current Role \n')
sns.countplot(x='years in current role', data=df)
plt.xlabel('\n No. of Working Year In current Role  ', fontsize = 10)
# plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  0 year - as we can also seen above that most the employees are fresher or having only 1 yea
#  2 years - highest no. of employees (372) are working since last 2 years in the current role
# 7 year - 2nd highest no. of employees (222) are working in a current role since 7 years, so
```



In [ ]:
```
```

In [196]:
```
#  33) Analysing Years Since Last Promotion ======>>>>>
```

In [197]:
```
df['years since last promotion'].unique()
```

Out[197]:
```
array([ 0,  1,  3,  2,  7,  4,  8,  6,  5, 15,  9, 13, 12, 10, 11, 14],
      dtype=int64)
```

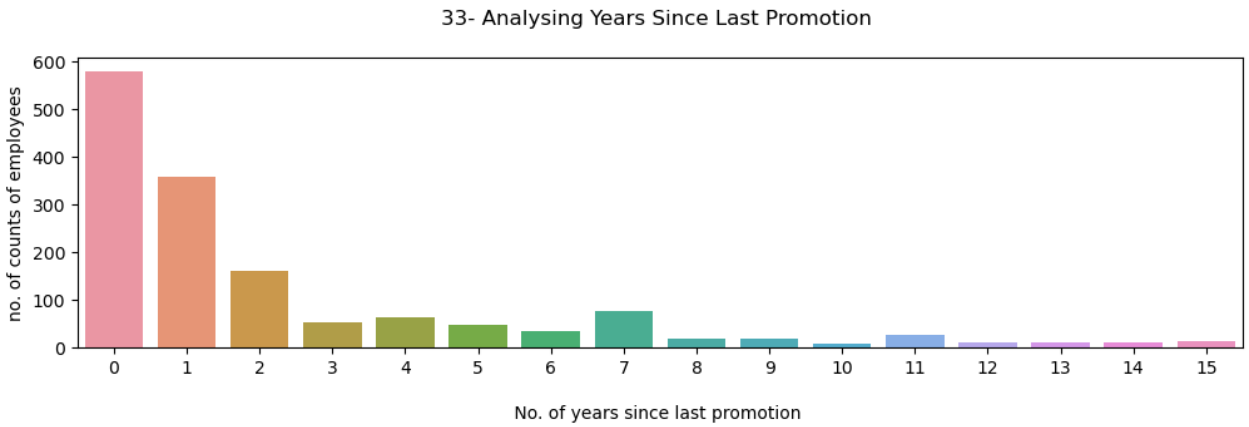
In [198]:
```
df['years since last promotion'].nunique()
```

Out[198]: 16

In [199]:
```
df['years since last promotion'].value_counts()
```

Out[199]:
```
0     581
1     357
2     159
7      76
4      61
3      52
5      45
6      32
11     24
8      18
9      17
15     13
13     10
12     10
14      9
10      6
Name: years since last promotion, dtype: int64
```

In [200]:
```
plt.figure(figsize = (12,3), facecolor='white')
plt.title('\n33- Analysing Years Since Last Promotion \n')
sns.countplot(x='years since last promotion', data=df)
plt.xlabel('\n No. of years since last promotion  ', fontsize = 10)
# plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  0 year - as we can also seen above that most the employees are fresher or having only 1 yea
#  1 years -2nd highest no. of employees (357) are working since last 1 years , so they may ge
# 2 year - 3nd highest no. of employees (159) are not get promoted since 2 years
#  15 years - there are also few of the employees are not get promoted since 15 years, they ma
```



In [ ]:

In [201]:
```
# 34) Analysing Number Of Years with current Manager =====>>>>
```

In [202]:
```
df['years with currmanager']
```

Out[202]:
```
0       5
1       7
2       0
3       0
4       2
       ..
1465    3
1466    7
1467    3
1468    8
1469    2
Name: years with currmanager, Length: 1470, dtype: int64
```

In [203]:
```
df['years with currmanager'].unique()
```

Out[203]:
```
array([ 5,  7,  0,  2,  6,  8,  3, 11, 17,  1,  4, 12,  9, 10, 15, 13, 16,
       14], dtype=int64)
```

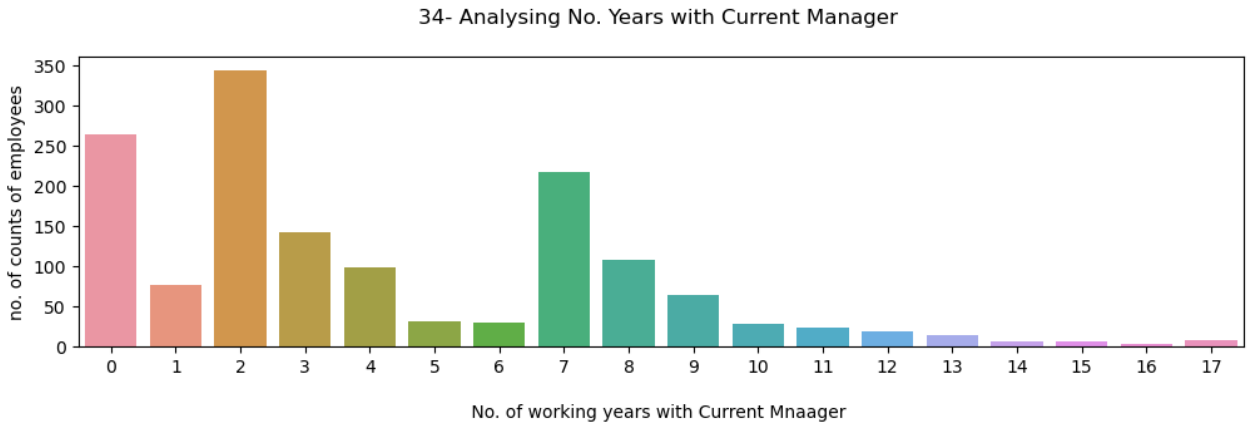
In [204]:
```
df['years with currmanager'].nunique()
#  there are 18 numbers of unique values are present inside the column.
```

Out[204]:
```
18
```

In [205]:
```
df['years with currmanager'].value_counts()
```

Out[205]:
```
2     344
0     263
7     216
3     142
8     107
4      98
1      76
9      64
5      31
6      29
10     27
11     22
12     18
13     14
17      7
15      5
14      5
16      2
Name: years with currmanager, dtype: int64
```

In [206]:
```
plt.figure(figsize = (12,3), facecolor='white')
plt.title('\n34- Analysing No. Years with Current Manager \n')
sns.countplot(x='years with currmanager', data=df)
plt.xlabel('\n No. of working years with Current Mnaager  ', fontsize = 10)
# plt.xticks(rotation=90,ha ='right')
plt.ylabel('no. of counts of employees', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  0 year - as we can also seen above that most the employees are fresher or having only less t
#  2 years -2nd highest no. of employees (263) are working since last 2 years with current mana
# 7 year - 3nd highest no. of employees (216) are working with current manager since 7 years. n
```



In [ ]:

========== HERE UNIVARIATE ANALYSIS IS COMPLETED FOR EACH COLUMN ==========

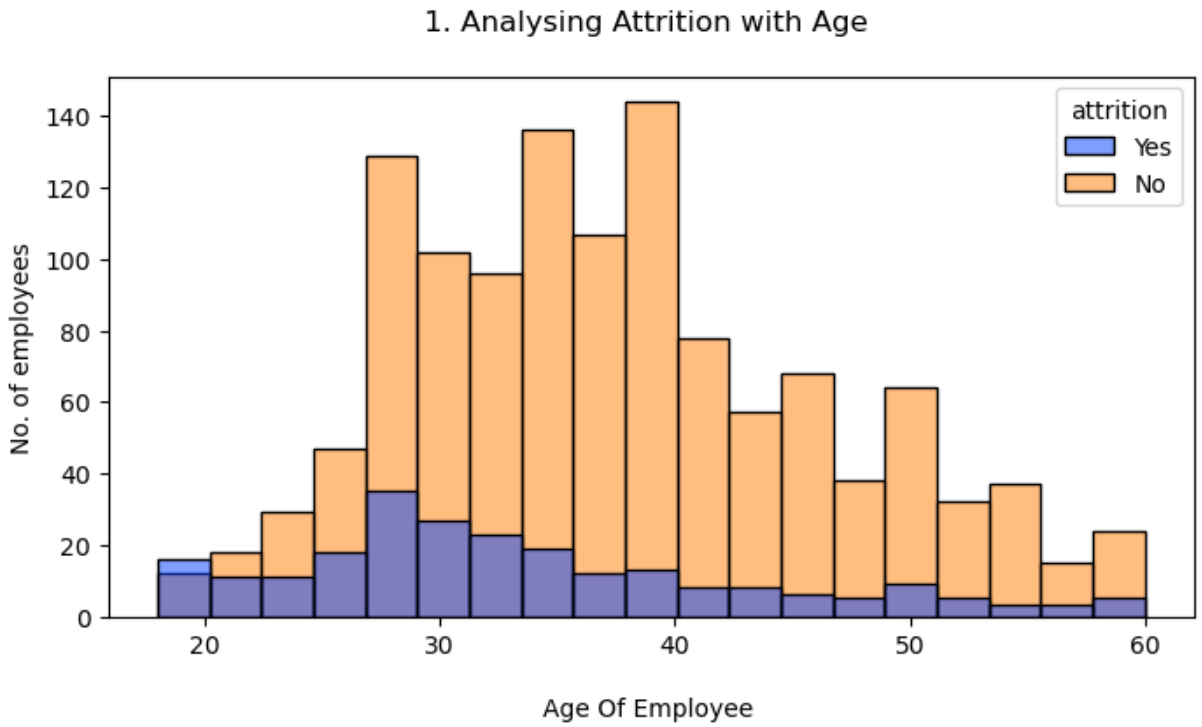
In [ ]:

======== TIME TO DO SOME BIVARIATE & MUTLIVARIATE ANALYSIS (to find ATTRITIION RATE)==========

In [207]:
```
# Here we wolud be  Analyse most of the RELEVANT COLUMNS WITH ATTRITION =====>
```

In [208]:
```
# 1) Analysing ATTRITION with AGE ====>>>
```

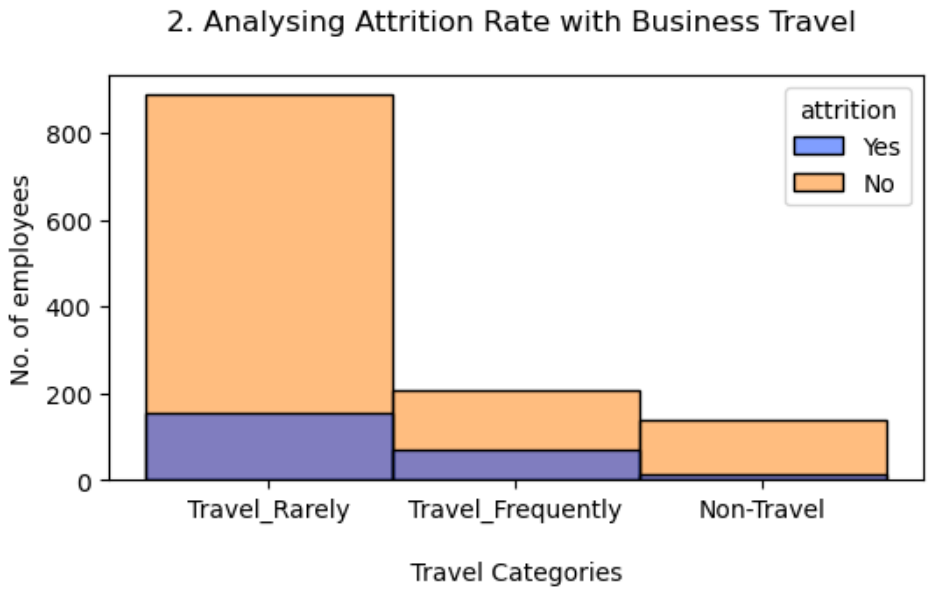
In [209]:
```
plt.figure (figsize = (8,4), facecolor = "white")
plt.title('\n1. Analysing Attrition with Age \n')
sns.histplot (x= 'age', hue = 'attrition', data= df,palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nAge Of Employee ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we can clearly find that the ATTRITION RATE is HIGHER in the LOWER AGE EMPLOYEES as co
# the ATTRITION RATE IS HIGHEST near the 30 year age of employees.
# from this we can conclude that the yong age employees are frequently switiching their compan
# ..or in the search of new job role.
# the HIGHER AGE EMPLOYEES are switichingless beaucse they may want STABILITY in their job.
```



In [ ]:

In [210]:
```
#  2) Analysing Attrition with Business travel ====>>>
```

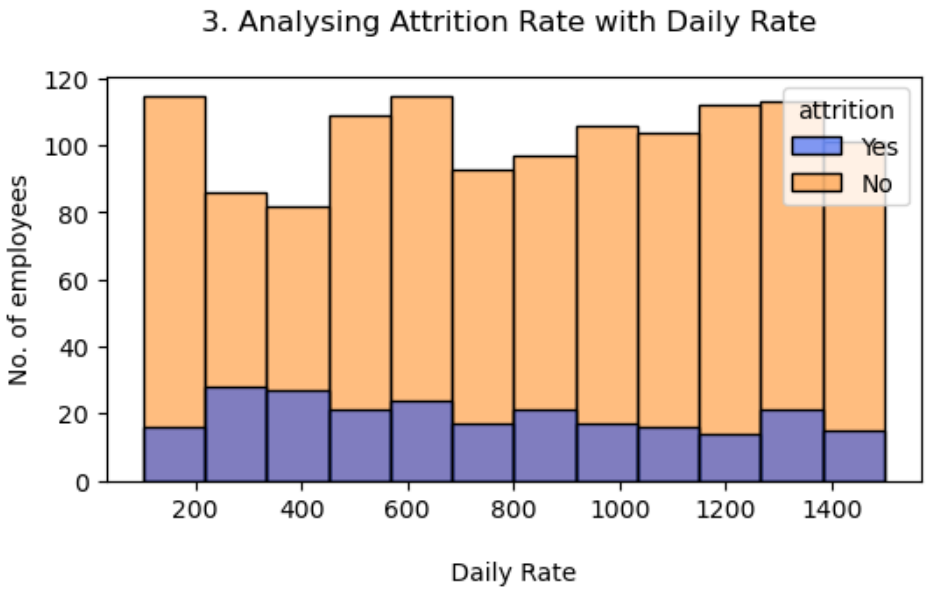
In [211]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n2. Analysing Attrition Rate with Business Travel \n')
sns.histplot (x= 'business travel', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Travel Categories ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that :
#  1- the Attrition Rate is Higher in those employees who 'Travel_Rarely'
# 2- the ATTRITION rate is LOWEST in NON-TRAVELER EMPLOYEES
```



In [ ]:
```
```

In [212]:
```
#  3) Analysing Attrition with Daily Rate ======>>>>
```

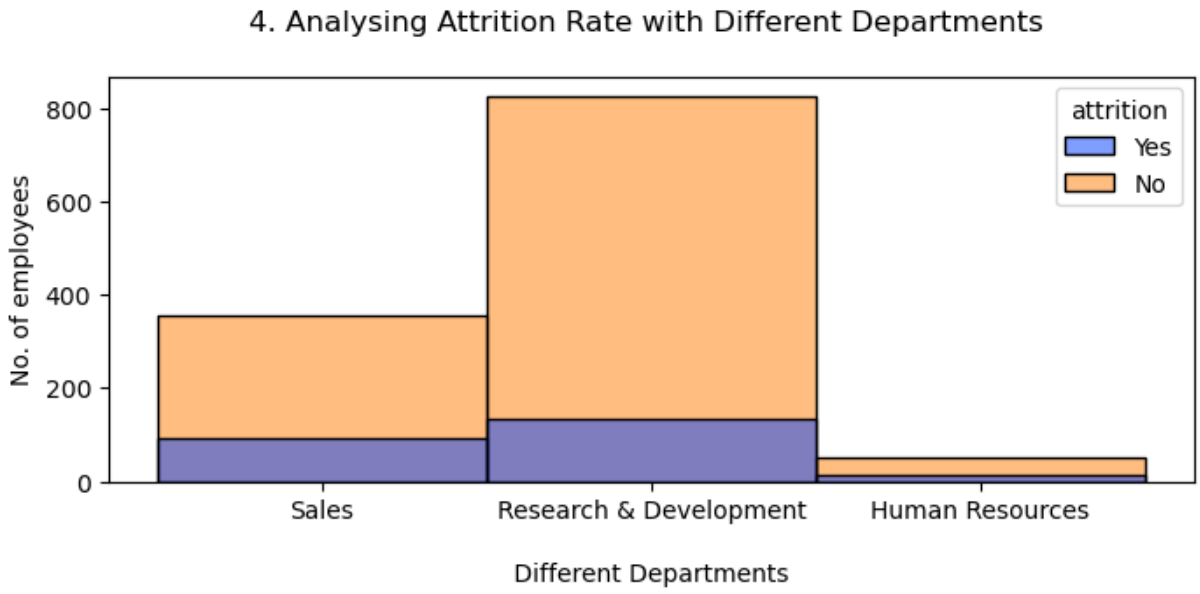
In [213]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n3. Analysing Attrition Rate with Daily Rate \n')
sns.histplot (x= 'daily rate', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Daily Rate ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that there is quit little difference in the ATTRITION RA
#  those employees who are getting less daily rate , the attrition is higher .
#  as compared with the attrition rate of those employees who are getting higher daily rate.
```



In [ ]:
```
```

In [214]:
```
# 4) Analysing ATTRITION in different DEPARTMENTS. ======>>>>
```

In [215]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n4. Analysing Attrition Rate with Different Departments \n')
sns.histplot (x= 'department', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Different Departments ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below bargraph we can find that the HIGHEST ATTRITION is in RESEARCH & DEVELOPEMI
# ...because this department is having highest no. of employees that's why it is showing highes
```

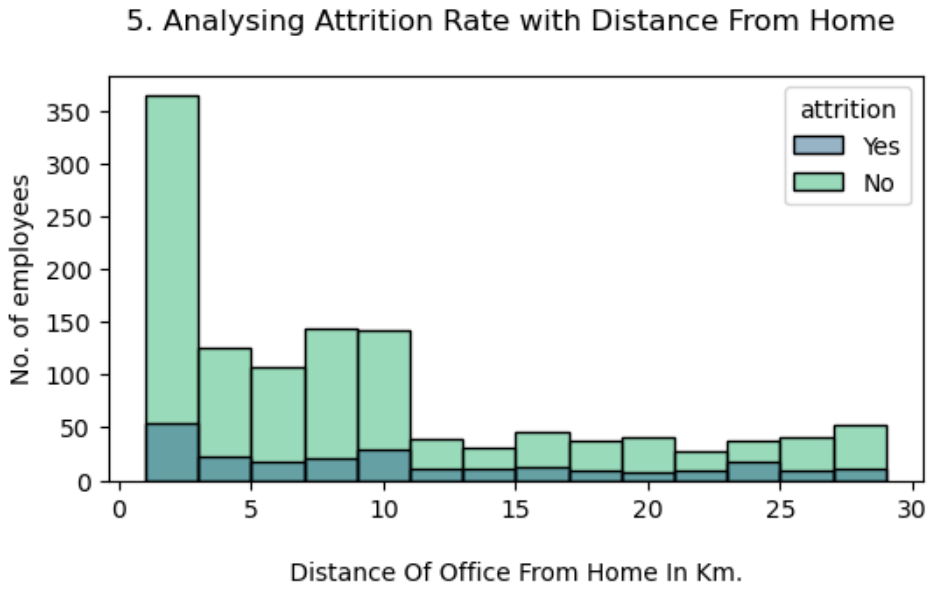


In [ ]:
```

```

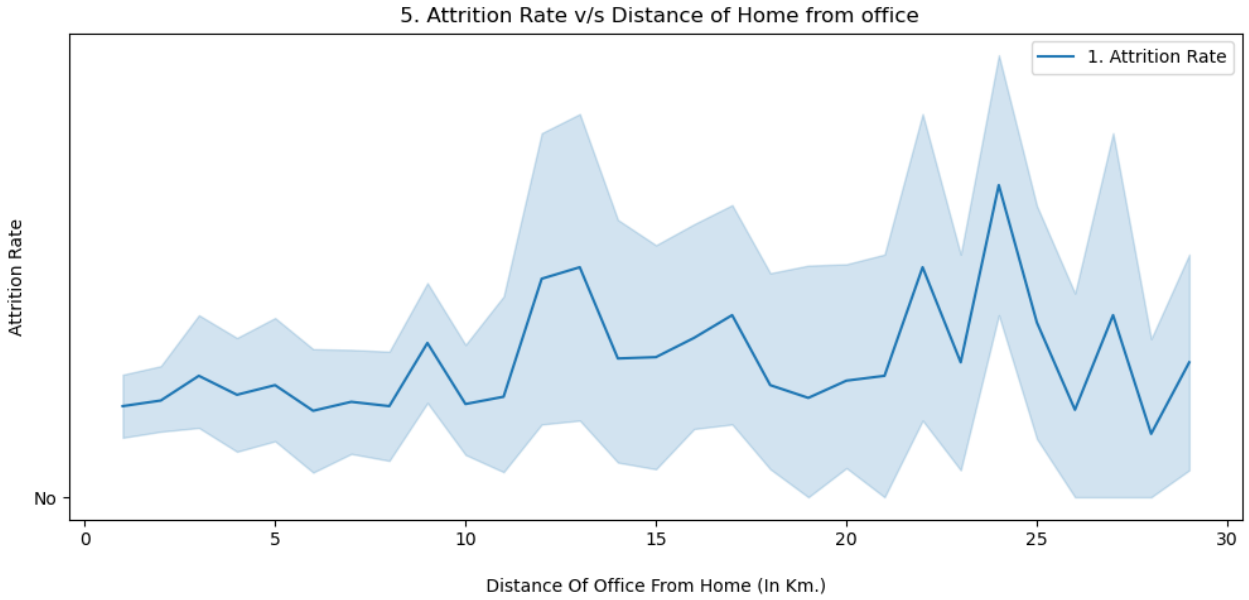
In [216]:
```
#  5) Analysing Attrition with Distance From Home. ======>>>>
```

In [217]:
```
# here below we trying to find the attrition rate with distance of office from employees houses
```

In [218]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n5. Analysing Attrition Rate with Distance From Home \n')
sns.histplot (x= 'distance from home', hue = 'attrition', data= df, palette = "viridis")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Distance Of Office From Home In Km. ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that the Highest ATTRITION RATE is in 0-3 km, beause the
# ..living in aournd that distance.
# but if we can see minutly then we can find the [ % of TOTAL EMPLOYEES AT THAT DISTANCE WITH A
# ...see the % of ATTRITION is HIGHER , near at 10 km distance. (weather the total employees at
# there is also HIGH ATTRITATION at 25 km distance also.
# by above analysis we can say that...

# CONCLUSION =>   [ HIGHER THE DISTANCE OF OFFICE = HIGHER THE ATTRITION RATE ]
```

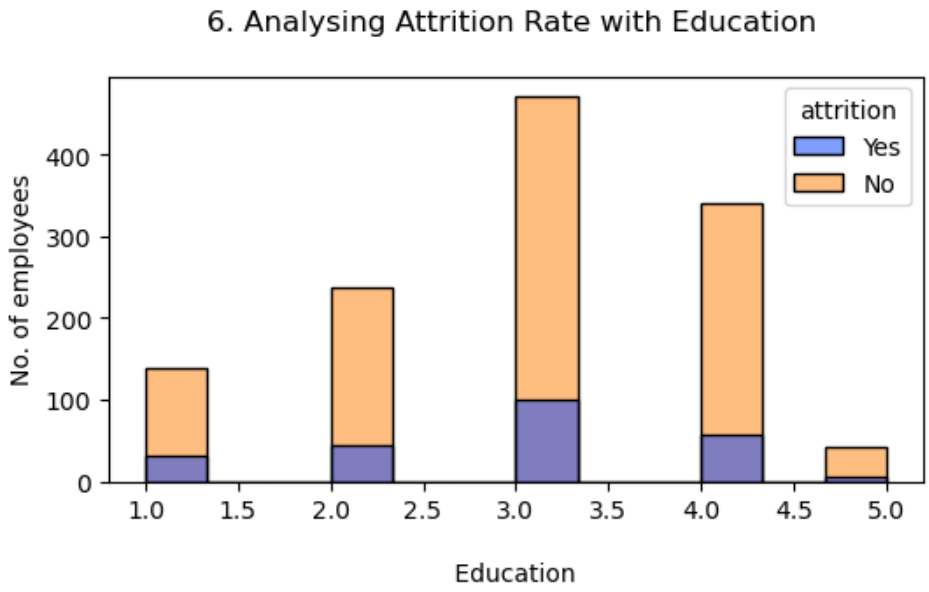


In [219]:
```
plt.figure(figsize=(12,5))
plt.title('\n 5. Attrition Rate v/s Distance of Home from office')
sns.lineplot (data=df, x='distance from home', y='attrition', label = '1. Attrition Rate')
plt.xlabel('\n Distance Of Office From Home (In Km.) ',fontsize=10)
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('Attrition Rate',fontsize=10)
# plt.yticks (rotation = 0, ha='center')
plt.show()

# MORE THE DISTANCE OF OFFICE FROM HOME = HIGHER ATTRITION RATE
```



In [220]:
```
#  6) Analysing Attrition Rate With Education ===>>>>
```

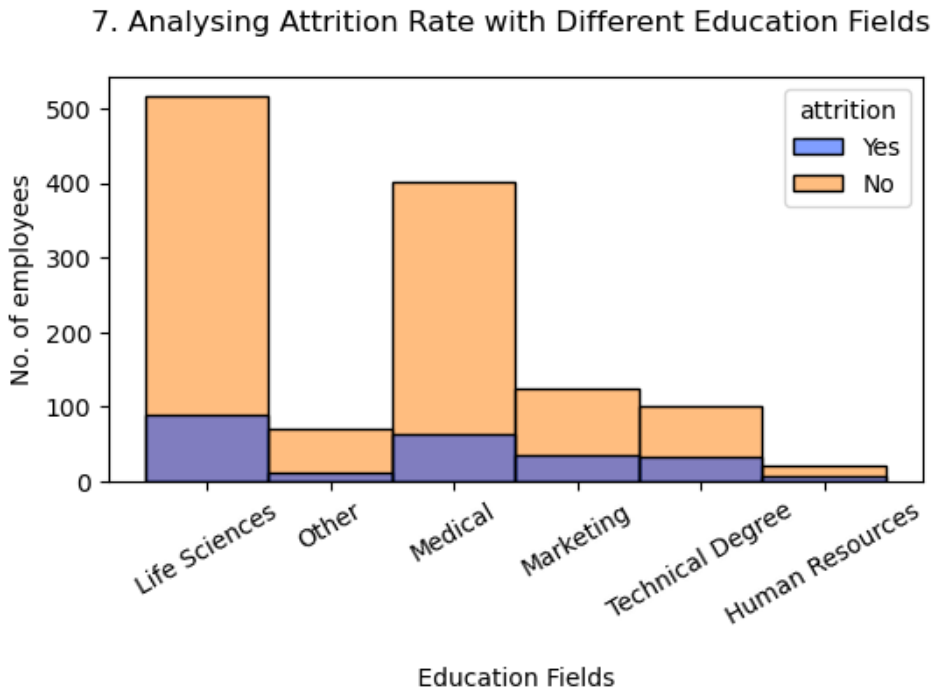
In [221]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n6. Analysing Attrition Rate with Education \n')
sns.histplot (x= 'education', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Education ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that ATTRITION RATE is HIGHEST in EDUCATION CODE - 3
#                                                    2nd HIGHEST in EDUCSTON CODE - 4
```



In [ ]:
```
```

In [222]:
```
# 7) Analysing Attrition Rate in Education Field ======>>>>
```

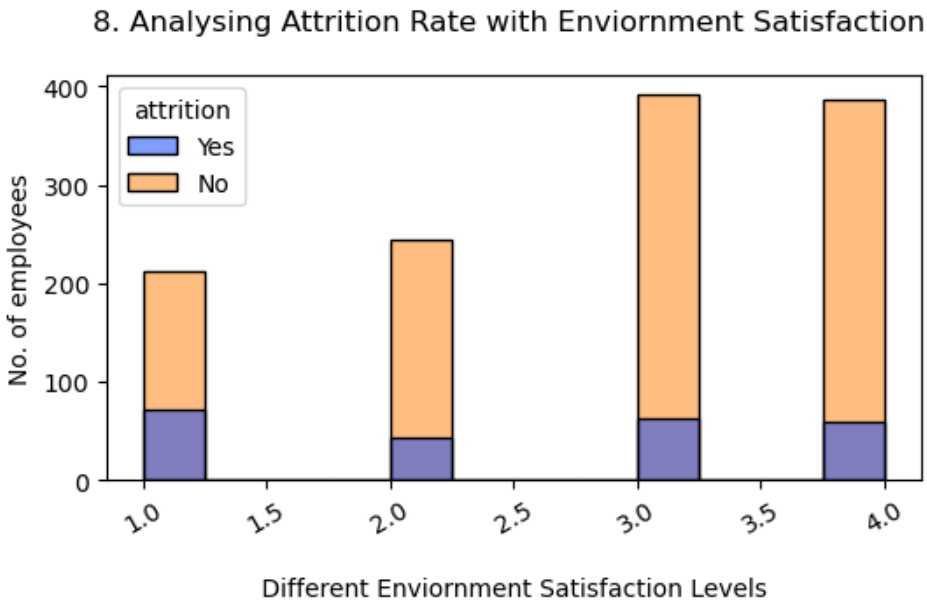
In [223]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n7. Analysing Attrition Rate with Different Education Fields \n')
sns.histplot (x= 'education field', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Education Fields ')
plt.xticks (rotation = 30, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that ATTRITION RATE is HIGHEST in LIFE SCIENCE AND MEDICA
# but if we can Minutely obderve the % of attritions, then ATTRITION IN 'MEDICAL' is Higher as

# CONCLUSION ==>   HIGHER ATTRITION RATE IN = LIFE SCIENCE & MEDICAL (Higher % in MEDICAL)
```



In [ ]:
```

```

In [224]:
```
#  8) Analysing Attrition rate with Enviornment Satisfaction =====>>>
```

In [225]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n8. Analysing Attrition Rate with Enviornment Satisfaction \n')
sns.histplot (x= 'environment satisfaction', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Different Enviornment Satisfaction Levels ')
plt.xticks (rotation = 30, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that ATTRITION RATE is HIGHEST in code- 1
# .. because the ration of ATTRITION in code-1 , as compared to TOTAL EMPLOYEE OF CODE-1 is Hig
# here in the code-1 , i think those employees are present who gives RATING-1 (LOWEST) to the C
#  therefore the ATTRITION is also HIGHER in this code-1

# CONCLUSION ==> LOWER THE ENVIORNMENT SATISFACTION = HIGHER THE ATTRITATION OF EMPLOYESS
```

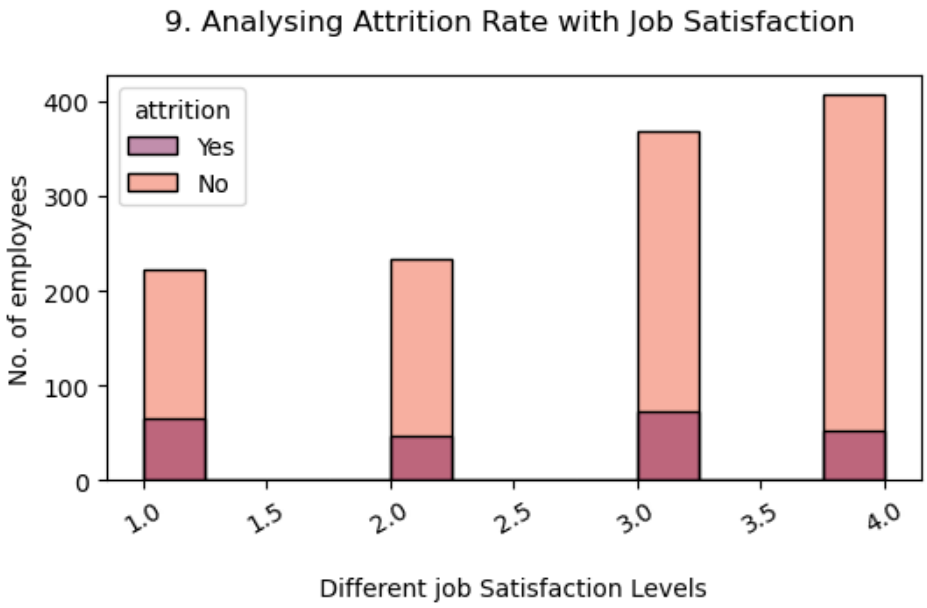


In [ ]:

In [226]:
```
#  9) Analysing Attritation rate with Job Satisfaction ===>>
```

In [227]:
```
#  job satisfaction rating are given by the employees to their current job.
```

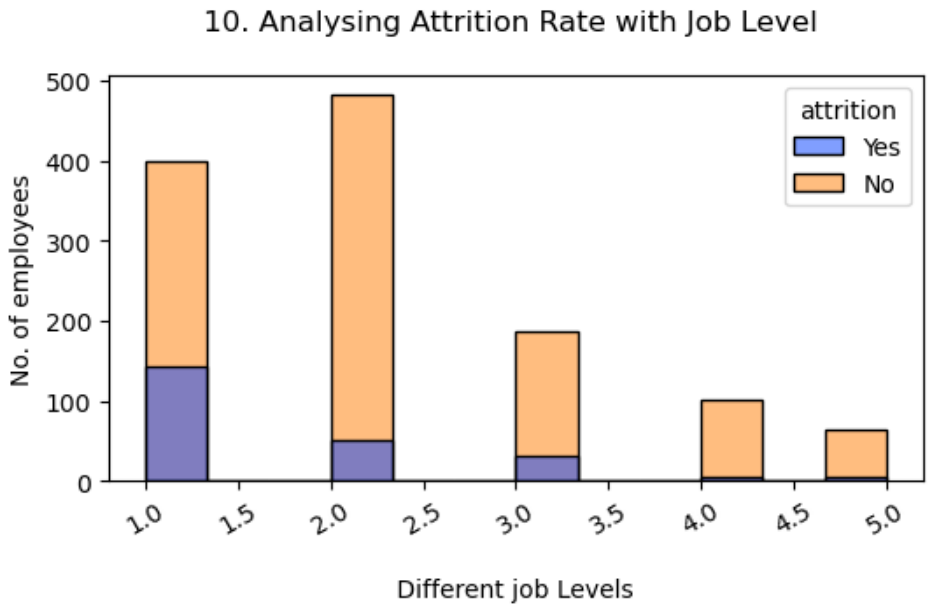
In [228]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n9. Analysing Attrition Rate with Job Satisfaction \n')
sns.histplot (x= 'job satisfaction', hue = 'attrition', data= df, palette = "rocket")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Different job Satisfaction Levels ')
plt.xticks (rotation = 30, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that ATTRITION RATE is HIGHEST in code- 1 & 3
# .. but the ration of ATTRITION in code-1 is higher then code-3, as compared to TOTAL EMPLOYEE
# here in the code-1 , those employees are present who gives RATING-1 (LOWEST) to their current
#  therefore the ATTRITION is also HIGHER in this code-1

# CONCLUSION ==> LOWER THE JOB SATISFACTION LEVEL = HIGHER THE ATTRITATION OF EMPLOYESS
```



In [ ]:

In [229]:
```
# 10) Analysong Attrition Rate with Job Level ====>>>
```

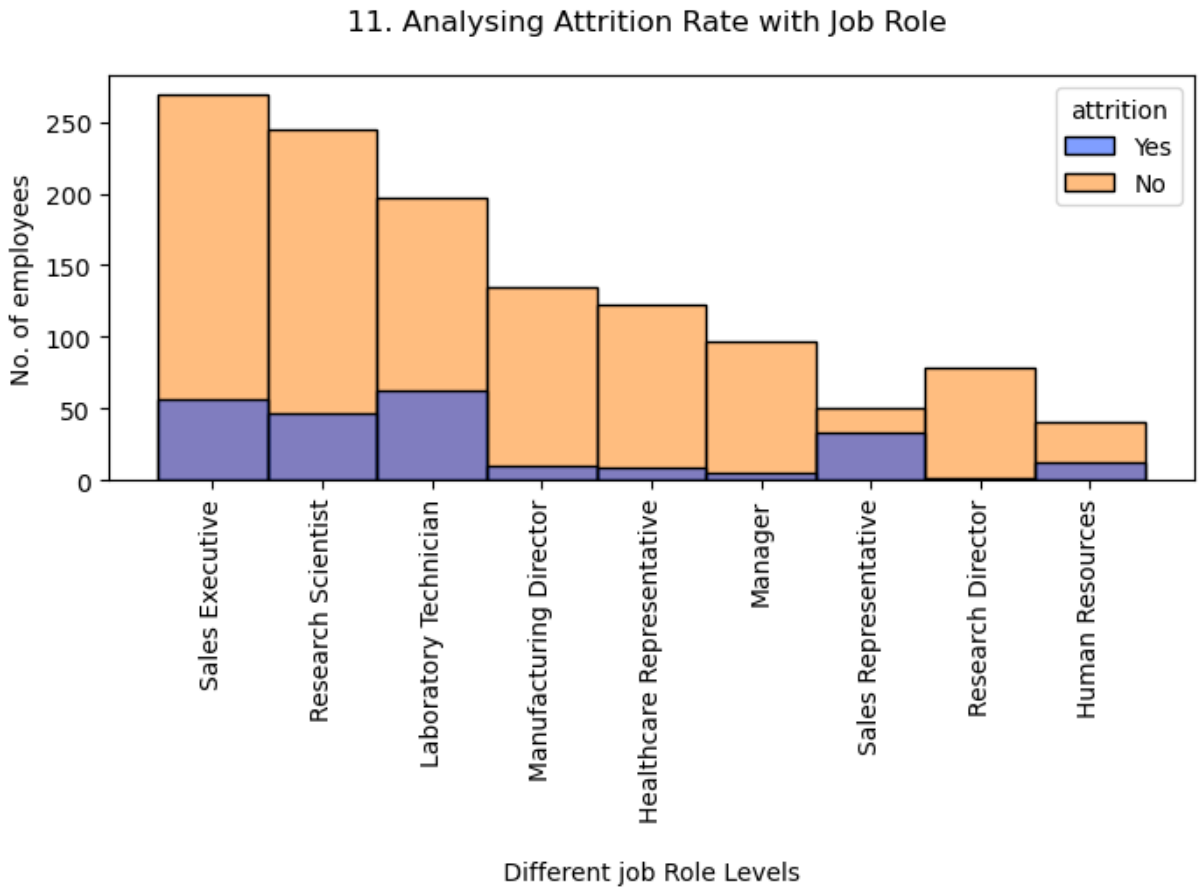
In [230]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n10. Analysing Attrition Rate with Job Level \n')
sns.histplot (x= 'job level', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Different job Levels ')
plt.xticks (rotation = 30, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can find the HIGHEST ATTRITION RATE is with LEVEL-1 JOBS
# Lowest attrition is on LEVEL-5 JOB'S

# CONCLUSION ==> LHIGHER THE ATTRITATION OF THOSE EMPLOYESS WHO ARE ON LEVEL-1 JOB (may Freshe
```



In [ ]:
```
```

In [231]:
```
# 11) Analysing Attrition Rate with Job Role ====>>>
```

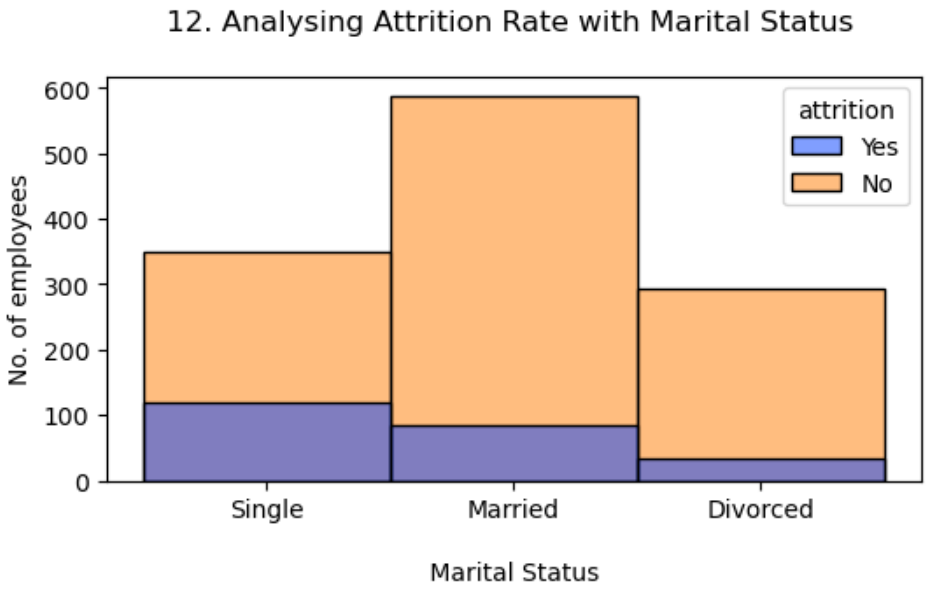
In [232]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n11. Analysing Attrition Rate with Job Role \n')
sns.histplot (x= 'job role', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Different job Role Levels ')
plt.xticks (rotation = 90, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that ATTRITION RATE is HIGHEST in LABORATORY TECHNISHIAN
#  but the INTRESTING FACT IS = the RATION OF ATTRITATION IS HIGHEST IN = SALES REPRESENTATIVE

# CONCLUSION ==>
# HIGHER THE ATTRITATION OF EMPLOYESS in 1) Sales Representative 2) Laboratory Technishian 3) S
```



In [ ]:

In [233]:
```
#  12) Analysing Attritation Rate in Marital Status :--
```

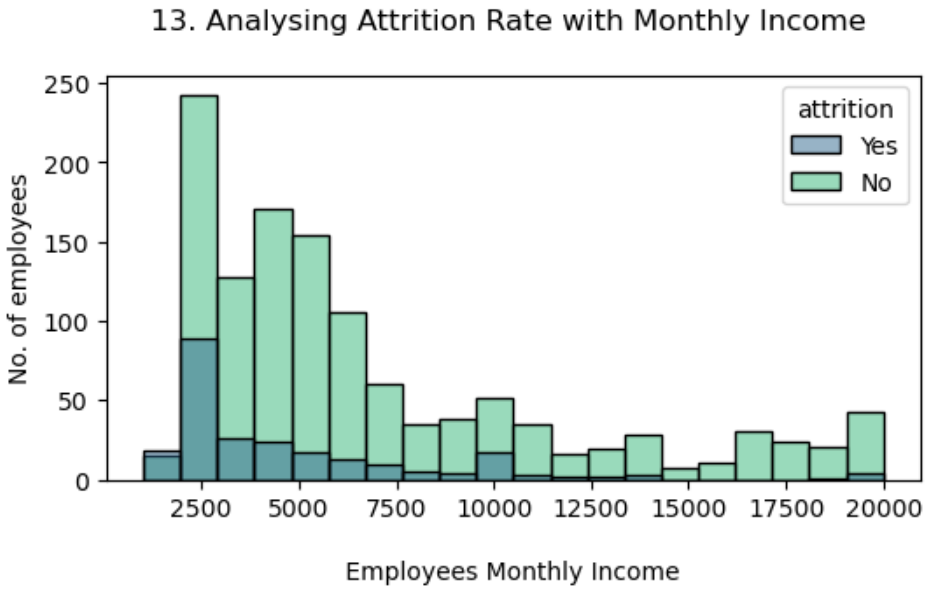
In [234]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n12. Analysing Attrition Rate with Marital Status \n')
sns.histplot (x= 'marital status', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Marital Status ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can see that the NON-MARRIED (SINGLE) employees are more Attrating as compared to MA
# CONCLUSION = HIGHEST ATTRATION IN 'SINGLE' EMPLOYEES
#              LOWEST IN DIVORCED
```



In [ ]:
```
```

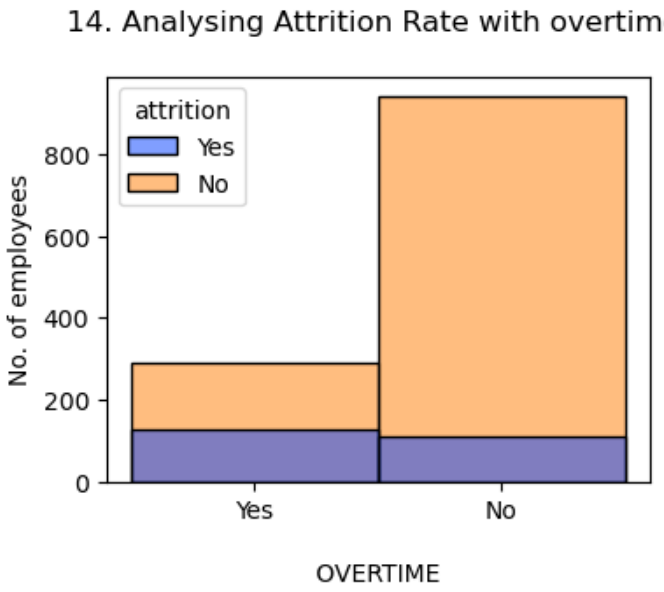
In [235]:
```
# 13) Analysing Attritation rate with Monthly income
```

In [236]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n13. Analysing Attrition Rate with Monthly Income \n')
sns.histplot (x= 'monthly income', hue = 'attrition', data= df, palette = "viridis")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Employees Monthly Income ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can see that the attritation rate is higher with Lower Monthly Salary getting employe
# there is a small attritation in HIGHEST PACKAGE GETTING EMPLOYES ALSO, that means HIGHER MANA
# ...the current job due to some reasons...

# CONCLUSION = HIGHEST ATTRATION  = LOWEST MONTHLY INCOME
#              they may switching company to get some higher package
#
```



In [237]:
```
# 14) Analysing Attritation Rate with OverTime ====>>>>>
```

In [238]:
```
plt.figure (figsize = (4,3), facecolor = "white")
plt.title('\n14. Analysing Attrition Rate with overtime \n')
sns.histplot (x= 'over time', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n OVERTIME ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can see that those employees are OVERTIMING  their attritation rate is verymuch
# ...Non-overtiming employees....
# offcourse we can predct that, the employees who is overtiming hey may need more money, and ij
#  ... then definatley he'll switch from current job

# CONCLUSION ==>  The Higher Attritation isin OVERTIMING EMPLOYEES can be seen.
```

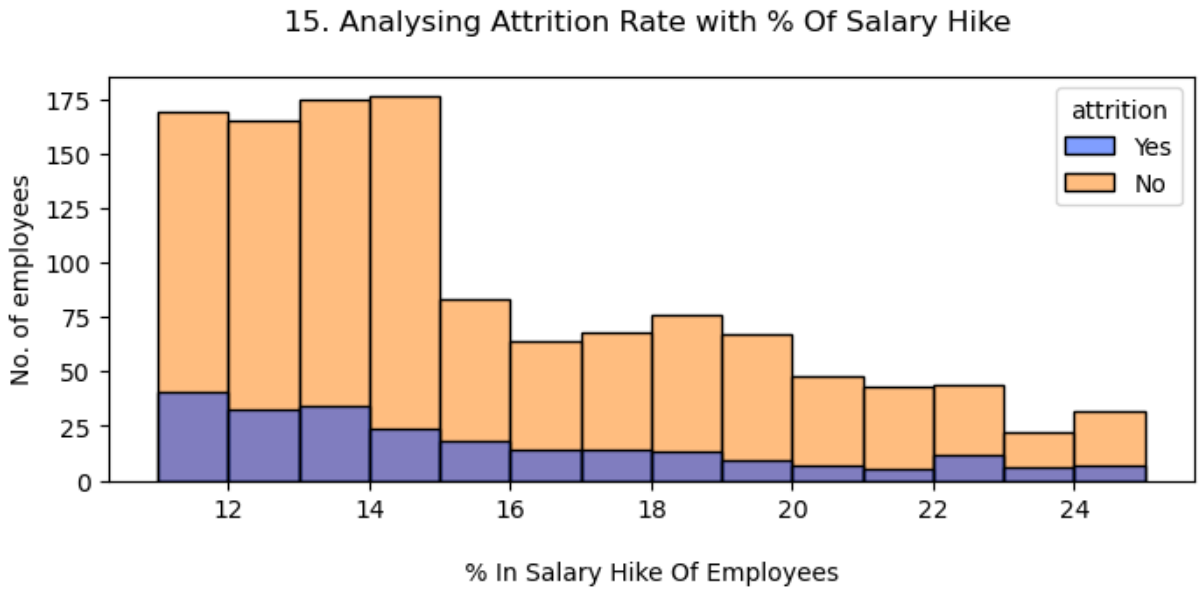


In [ ]:

In [ ]:

In [239]:
```
#  15) Analysing Attritaiton with Percent Salary Hike =====>>>
```

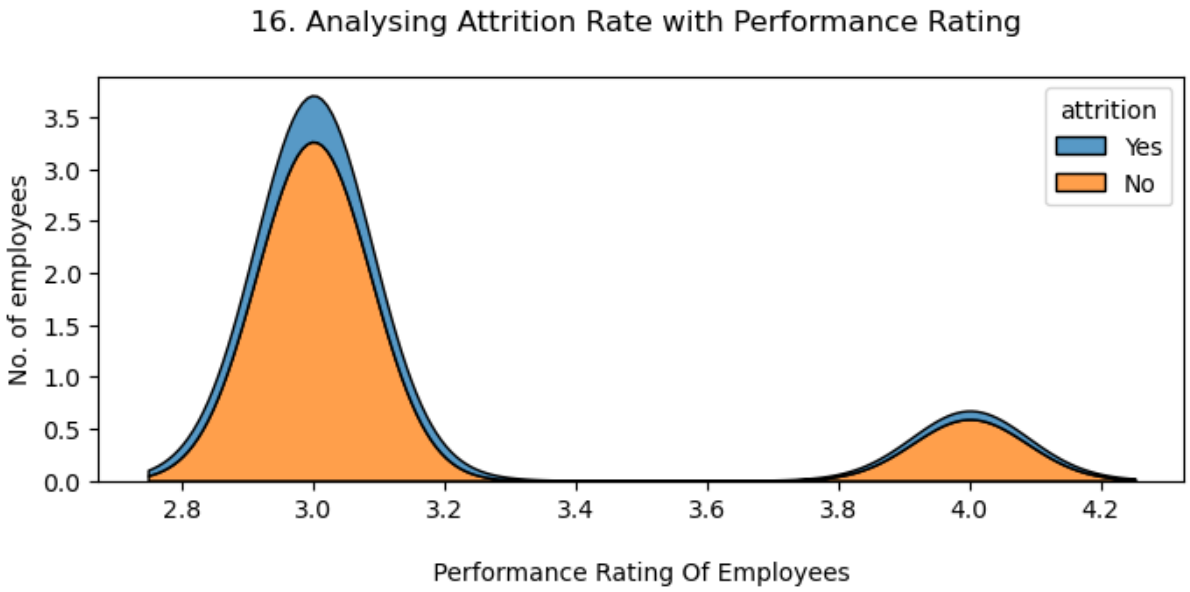
In [240]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n15. Analysing Attrition Rate with % Of Salary Hike \n')
sns.histplot (x= 'percent salary hike', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n % In Salary Hike Of Employees ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can find that Attritation Rate is DECREASING as the % Of Salary Hike is INCREAS
# the HIGHEST ATRRITATION can be seen with 10- 14 % Salary Hike

#  CONCLUSION = LOWER THE % IN SALARY HIKE = HIGHER ATTRITATION RATE
```



In [ ]:
```

```

In [241]:
```
#  16) Analysing Attritation Rate with Performance Rating =======>>
```

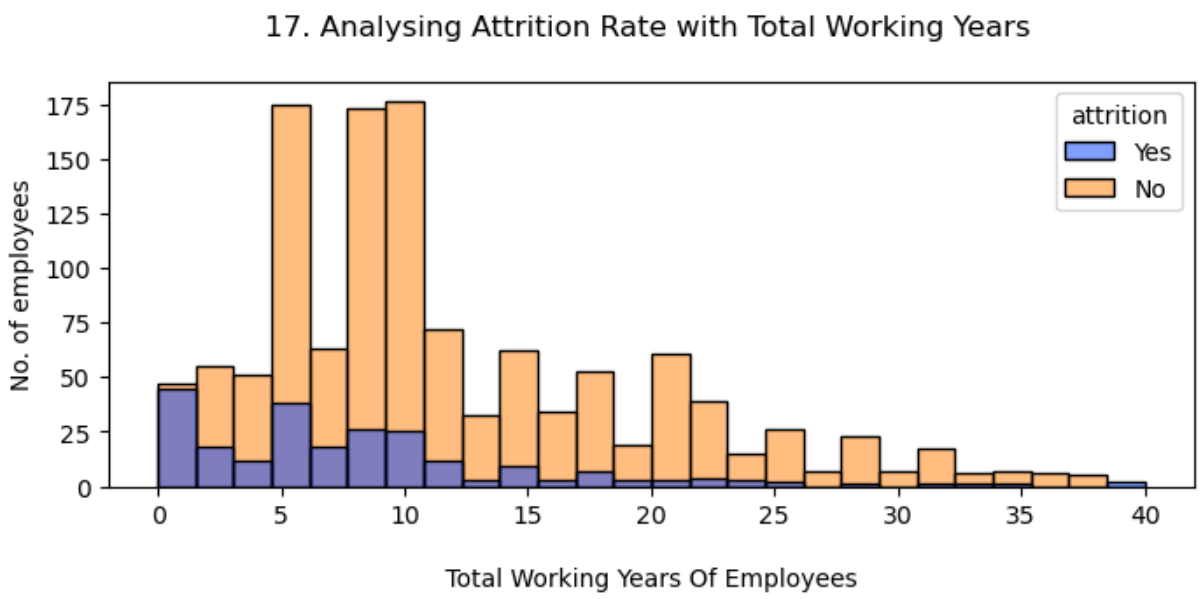
In [242]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n16. Analysing Attrition Rate with Performance Rating \n')
sns.kdeplot(data=df, x="performance rating", hue="attrition", multiple="stack")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Performance Rating Of Employees ')
# plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can find that Attritation Rate is HIgher with PERFORMANCE RATING-3

# CONCLUCSION = LOWER THE PERFORMANCE RATING = HIGHER THE ATTRITION
```



In [ ]:
```
```

In [243]:
```
#  17) Analysing Attrition with Total Working Years =====>>>
```

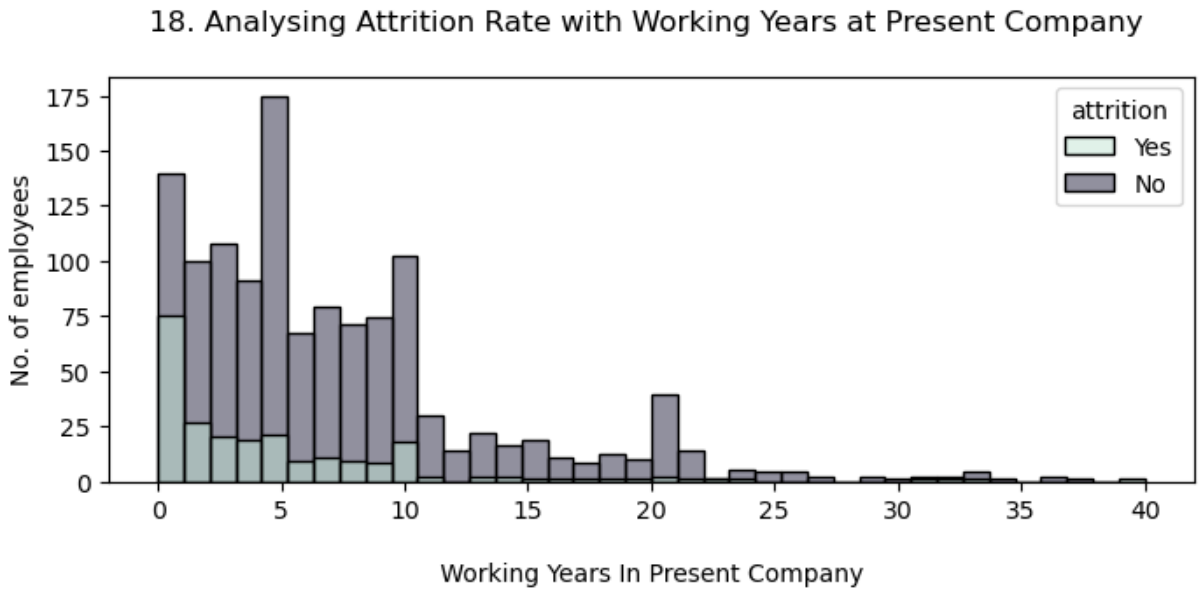
In [244]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n17. Analysing Attrition Rate with Total Working Years \n')
sns.histplot (x= 'total working years', hue = 'attrition', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Total Working Years Of Employees ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here in the below graph we can find that THE HIGHEST ATTRITION RATE is with 0-2 & 5-7 years w
# the ATTRITION RATIO is very LESS with MOST EXPERIANCED EMPLOYEES
# there is a small ATTRITION in 40 year experianced employees also, that means some HIGHER MANA

#  CONCLUSION = HIGHER ATTRITATION RATE WITH LOWER EXPERIANCE EMPLOYEES
```



In [ ]:

In [245]:
```
#  18) Analysing Attrition with Year's at Present Company ==============>>>>
```

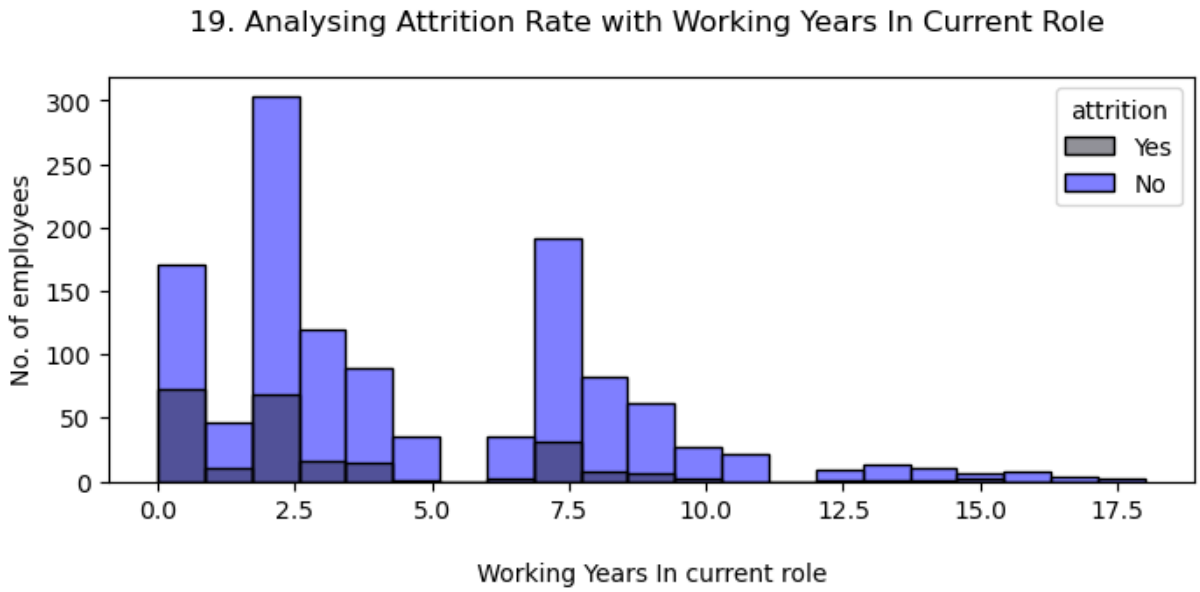
In [246]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n18. Analysing Attrition Rate with Working Years at Present Company \n')
sns.histplot (x= 'years at company', hue = 'attrition', data= df, palette = "ch:start=.2,rot=-
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Working Years In Present Company ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can find that Attritation Rate is HIGHEST with the LESS EXPERIANCED EMPLOYEES
# 0-1 YEAR EXPERIANCE in present company employees are more attriting as compared to more exper
# the attrition rate is HIGHER in 0-5 current experiance employees..

#  CONCLUSION = LOWER THE CURRENT EXPERIANCE = HIGHER ATTRITATION RATE
```



In [ ]:

In [247]:
```
# 19) Analysing Attrition Rate with Years In Current Role ======>>>
```

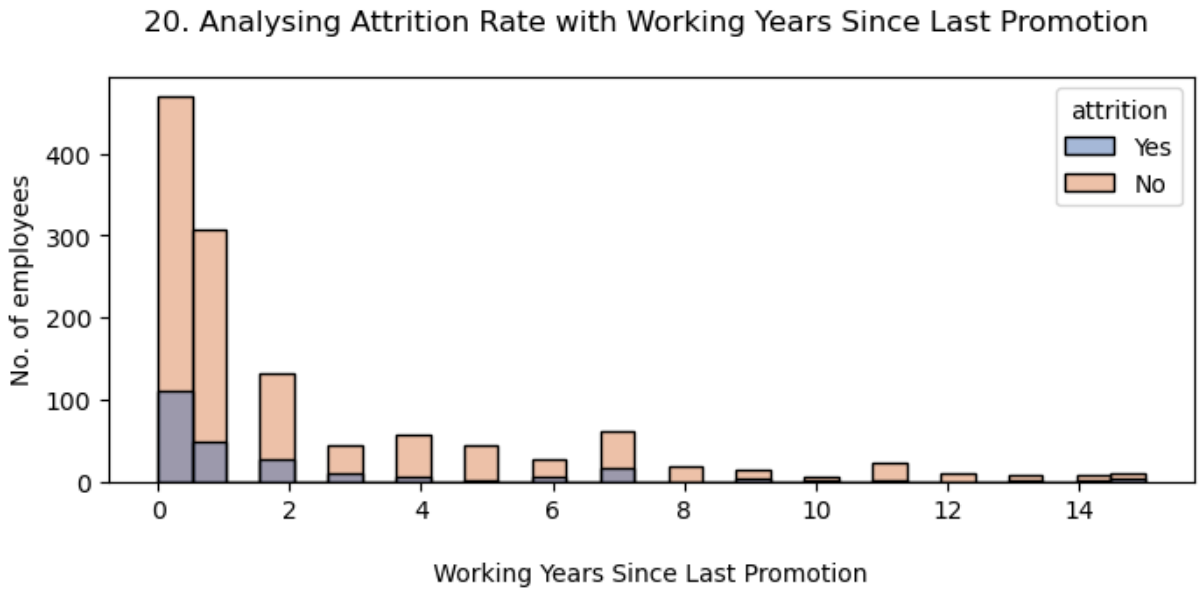
In [248]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n19. Analysing Attrition Rate with Working Years In Current Role \n')
sns.histplot (x= 'years in current role', hue = 'attrition', data= df, palette = "dark:blue")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Working Years In current role ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can find that Attritation Rate is HIGHEST with those employees who are having w
# some of the employees are also attritaing with working sinc 7 year in a same role

#  CONCLUSION = HIGHER ATTRITATION RATE = with 0-3 YEARS WORKING IN A SAME ROLE
```



In [ ]:
```
```

In [249]:
```
# 20) Analysing Attrition Rate with Working Years Since Last Promotion =====>>>>
```

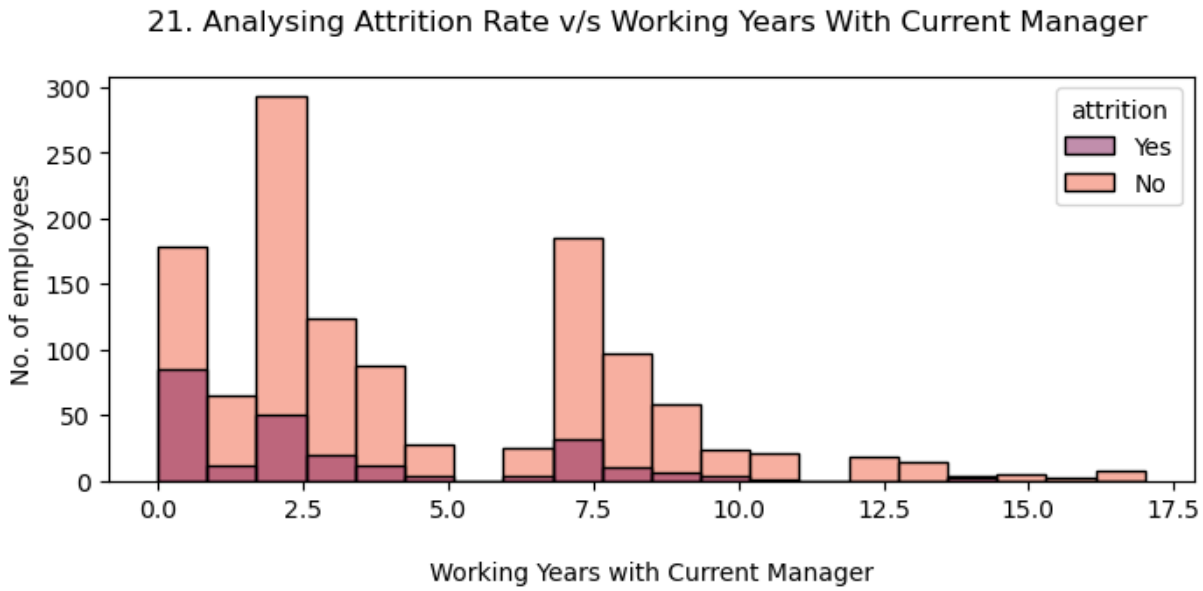
In [250]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n20. Analysing Attrition Rate with Working Years Since Last Promotion \n')
sns.histplot (x= 'years since last promotion', hue = 'attrition', data= df, palette = "deep")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Working Years Since Last Promotion ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can find that Attritation Rate is HIGHEST with those employees who are having 0
# ....since last promotion
# Most of the employees are switching campany within 0-1 year AFTER GETTING PROMOTED

#  CONCLUSION = HIGHER ATTRITATION RATE WITH = RECENTLY PROMOTED EMPLOYEES
```



In [ ]:

In [251]:
```
# 21) Analysing Attrition Rate v/s Years working with Current Manager ====>>
```

In [252]:
```
plt.figure (figsize = (8,3), facecolor = "white")
plt.title('\n21. Analysing Attrition Rate v/s Working Years With Current Manager \n')
sns.histplot (x= 'years with currmanager', hue = 'attrition', data= df, palette = "rocket")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\n Working Years with Current Manager ')
plt.xticks (rotation = 0, ha= 'center')
plt.ylabel('No. of employees')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here below we can find that Attritation Rate is HIGHEST with those employees who are working
# Most of the employees are switching campany within 0-1 year working with current manager alsc

#  CONCLUSION = HIGHER ATTRITATION RATE WITH = LESS WORKING YEARS WITH CURRENT MANAGER.
```



In [ ]:

========== BIVARIATE ANALYSIS COMPLETED
==========

In [ ]:

========== MULTI-VARIATE ANALYSIS
==========

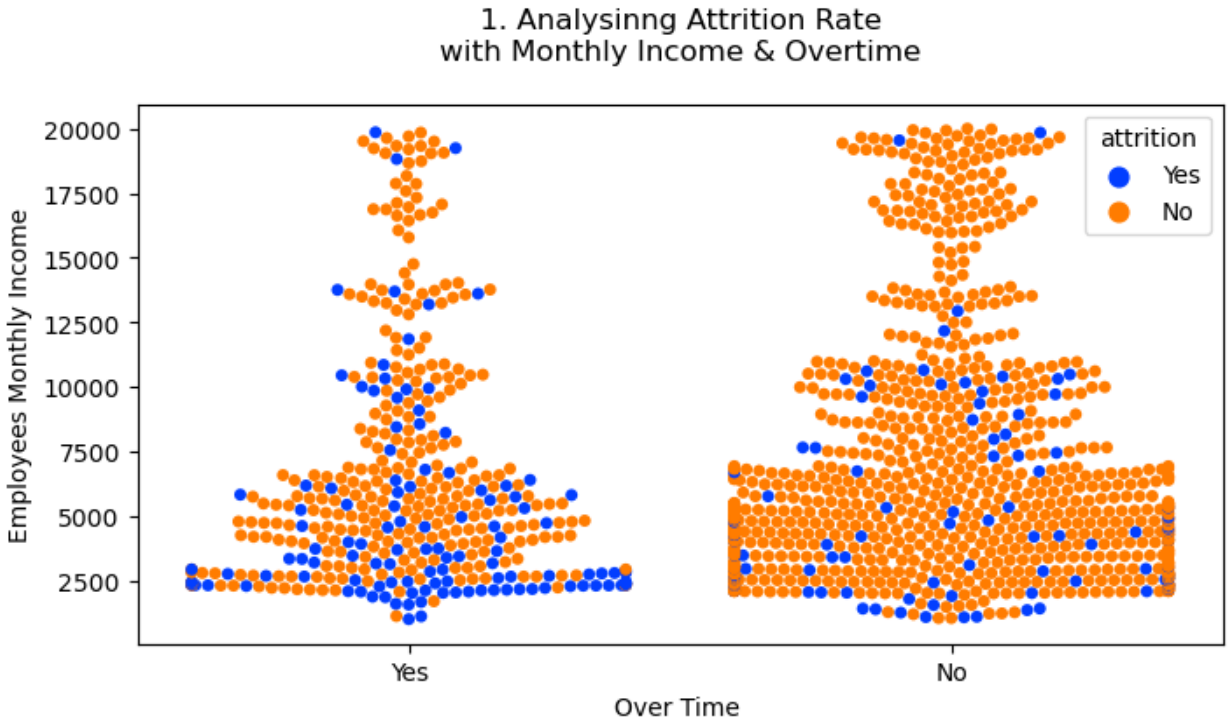
In [253]:
```
#  HERE IN THE MULTIVARIATE ANALYSIS WE CAN ANALYSE MORE THEN 2 DIMENSIONS WITH OUT TARGET (AT
```

In [ ]:

In [ ]:

In [254]:
```
#  1) Analysing Attrition with Monthly salary & Over Time =====>>>>
```

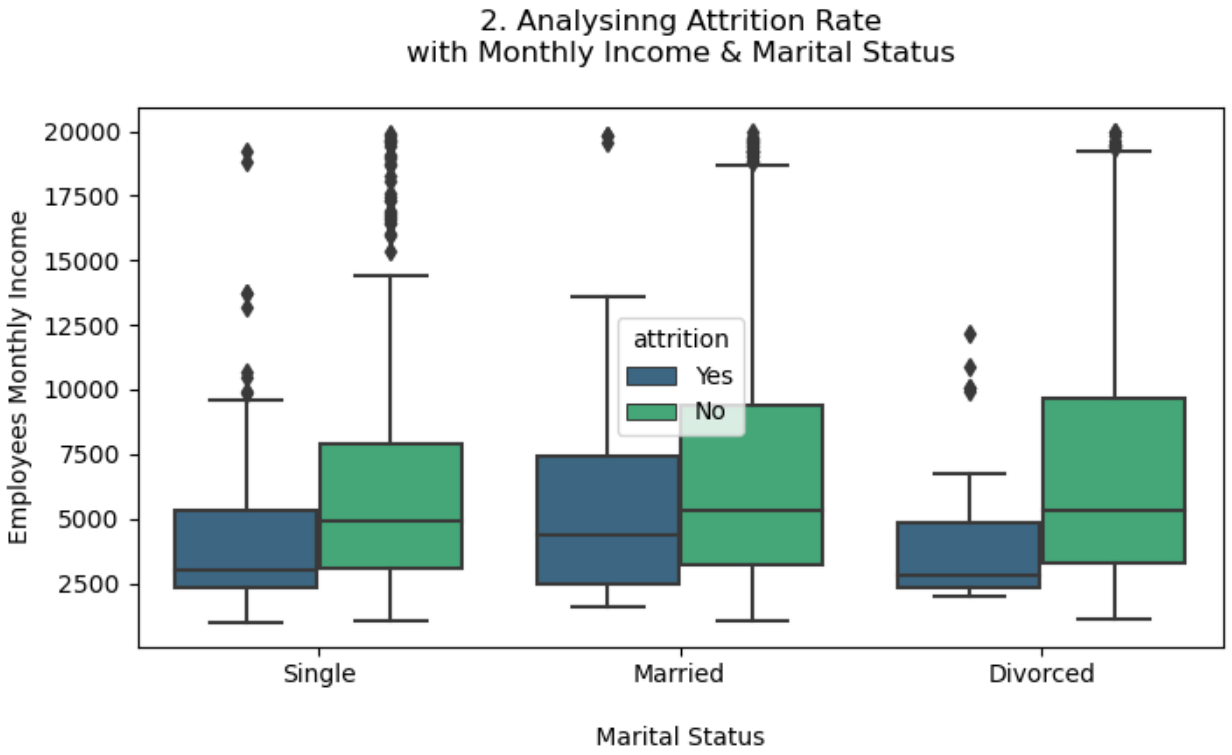
In [255]:
```
plt.figure(figsize=(8,4),facecolor="white")
plt.title('\n 1. Analysinng Attrition Rate \n with Monthly Income & Overtime \n')
sns.swarmplot (x= 'over time', y= 'monthly income',hue = 'attrition', data= df, palette = "bri
plt.xlabel ('Over Time ')
plt.xticks(rotation=0, ha='center',fontsize=10)
plt.ylabel('Employees Monthly Income')
plt.show()


# Here from the below SWARM PLOT we can find the following things :-
# 1- The Density of NO-OVERTIME Employees are HIGHER then the DENSITY OF YES-OVERTIME Employees
# 2- the DENSITY of NO-OVERTIME Employees is HIGHER in SALARY BETWEEN 2500 - 7500 $
# 3- The ATTRITION RATE is HIGHER in YES-OVERTIME Employees as compared to NO-OVERTIME Employee
#    that means the OverTimer employees in the LOWER SALARY RANGE are more ATTRITATING as compa
# 4- The Higher Salary level for both OVER TIMER & NON-OVERTIMER employees are almost same.
```



In [ ]:

In [256]:
```
# 2) Analysing Attrition with Monthly Income & Marital Status.
```

In [257]:
```
plt.figure(figsize=(8,4),facecolor="white")
plt.title('\n 2. Analysinng Attrition Rate \n with Monthly Income & Marital Status \n')
sns.boxplot (x= 'marital status', y= 'monthly income',hue = 'attrition', data= df, palette = "
plt.xlabel ('\n Marital Status ')
plt.xticks(rotation=0, ha='center',fontsize=10)
plt.ylabel('Employees Monthly Income')
# plt.legend(loc='center')
plt.show()


# Here from the below Box PLOT we can find the following things :-
# 1- here in all three Marital Status categorie we can clearly found that the NO-ATTRITION emp
#    ....YES-ATTRITION employees.
# 2- we found the HIGHER ATTRITION IN MARRIED EMPLOYEES with the salary range 2500-7500 $.
#    ....and also found some outliers also at $ 20,000 salary
# 3- 2nd Highest HIGHER ATTRITION RATE in SINGLE EMPLOYEES with SALARY RANGE of 2500-500 $
#    ....also found some attrition of outliers above $ 10,000.
# 4- the ATTRITION RATE is LOWEST in DIVORCED CATEGORY..
```

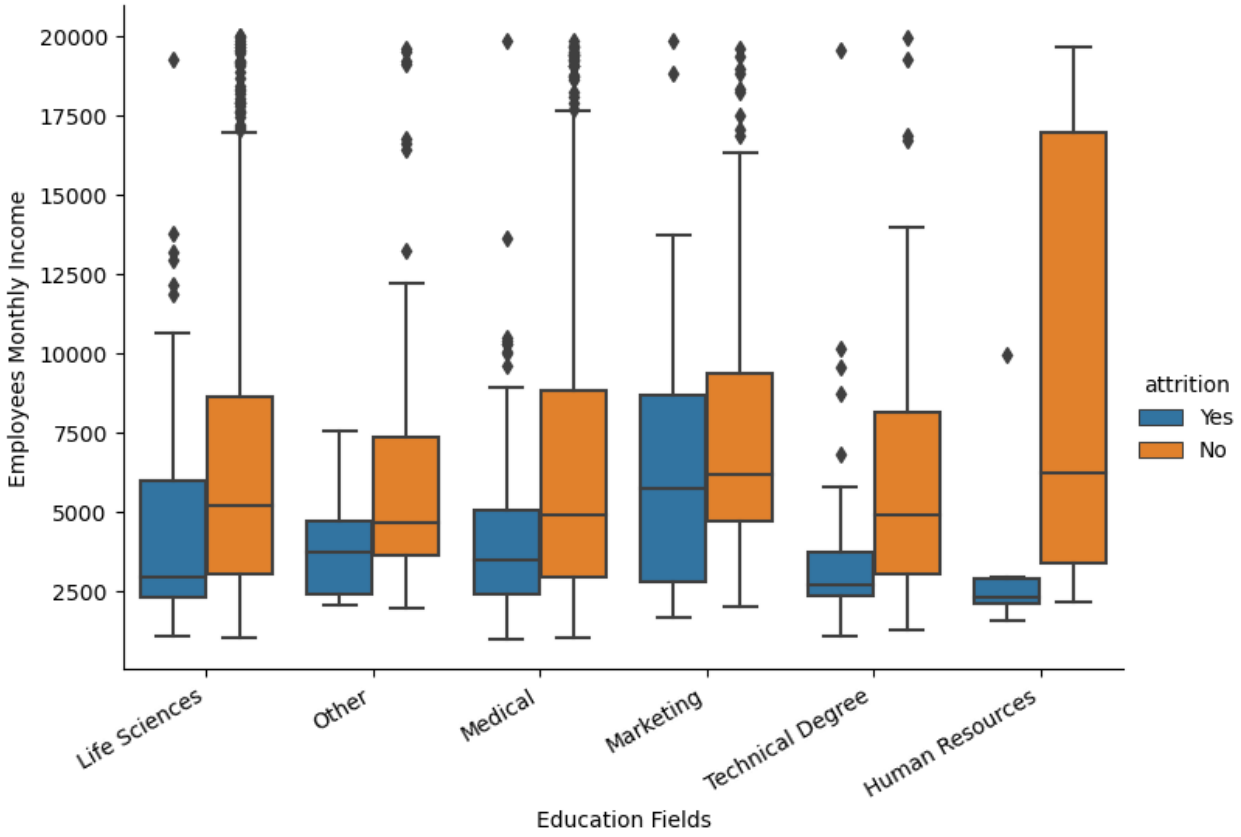


In [ ]:

In [258]:
```
# 3) Analysing Attrition with Monthly Income & Education Field ========>>>
```

In [259]:
```
plt.figure(figsize=(8,4),facecolor="white")
# plt.title('\n 3. Analysinng Attrition Rate \n with Monthly Income & education field \n')
sns.catplot(x="education field", y="monthly income", hue="attrition", data=df, kind="box", hei
plt.xlabel('Education Fields')
plt.xticks(rotation=30,ha='right')
plt.ylabel('Employees Monthly Income')
# plt.yticks(rotation=0,ha='center')
plt.show()

# here from the below BOX PLOT we can found that :
# 1- the Highest NON-ATTRITION in Human Resource Education Field
#    ...The Salary of Non-Attrition Employees Much Higher then the salary of YES- Attrition Emp

# 2- The Highest ATTRITION in Marketting and LIFE SCIENCE
#    here also we can find that the salary YES-ATTRITION employees are LESS THEN compared to NO
```

```
<Figure size 800x400 with 0 Axes>
```

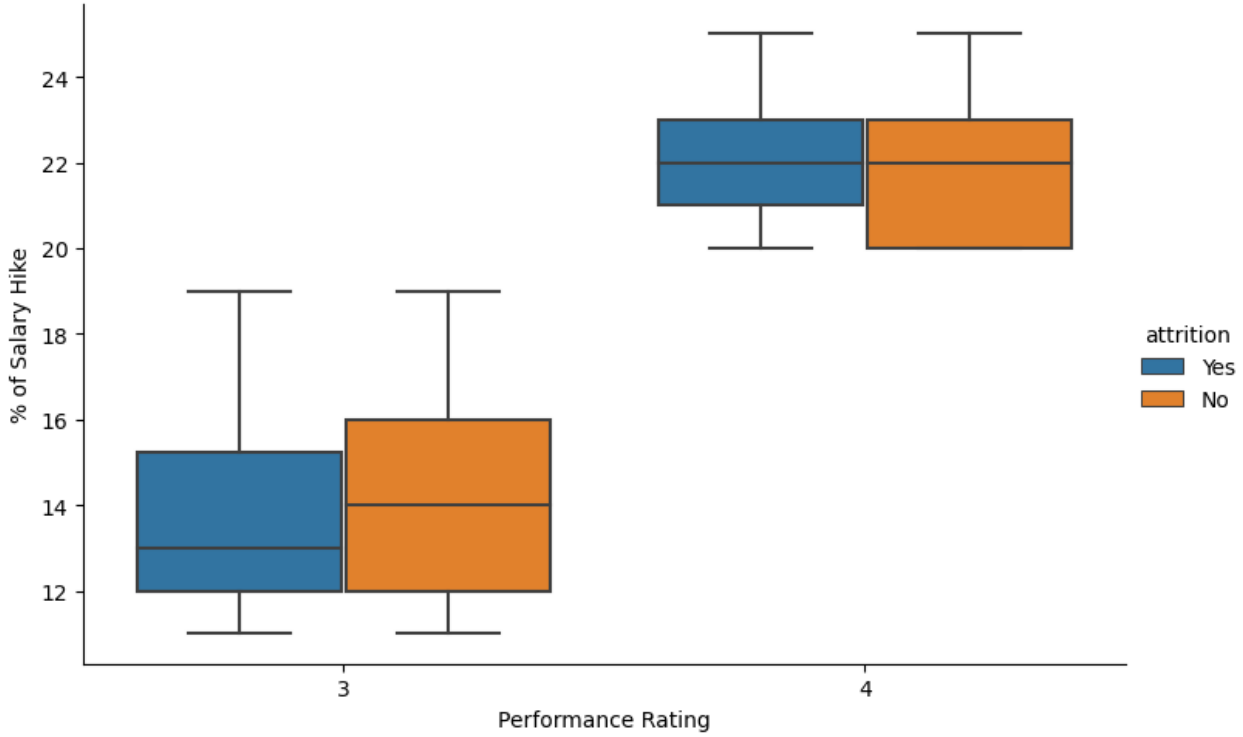


In [ ]:

In [260]:
```
# 4) Analysing Attrition Rate with Performance Rating & Percentage of Salary Hike ========>>>>
```

In [261]:
```
plt.figure(figsize=(6,3),facecolor="white")
# plt.title('\n 4. Analysinng Attrition Rate \n with Monthly Income & education field \n')
sns.catplot(x="performance rating", y="percent salary hike", hue="attrition", data=df, kind="b
plt.xlabel('Performance Rating')
# plt.xticks(rotation=30,ha='right')
plt.ylabel('% of Salary Hike')
# plt.yticks(rotation=0,ha='center')
plt.show()

# here below in the BOX PLOT it can be cleared that the PERFORMANCE RATING-3 (PR_3)  is DOWNGRA
# so we can see the clear difference that the ' % of Salary Hike' is verymuch Higher of PR-4 Th
# here we can also find that the TRRITION RATE  is Much Higher in PR-3 as compared to PR-4
# those employees whose % of salary hike is not equals to their coliuge, they may switiching th
```

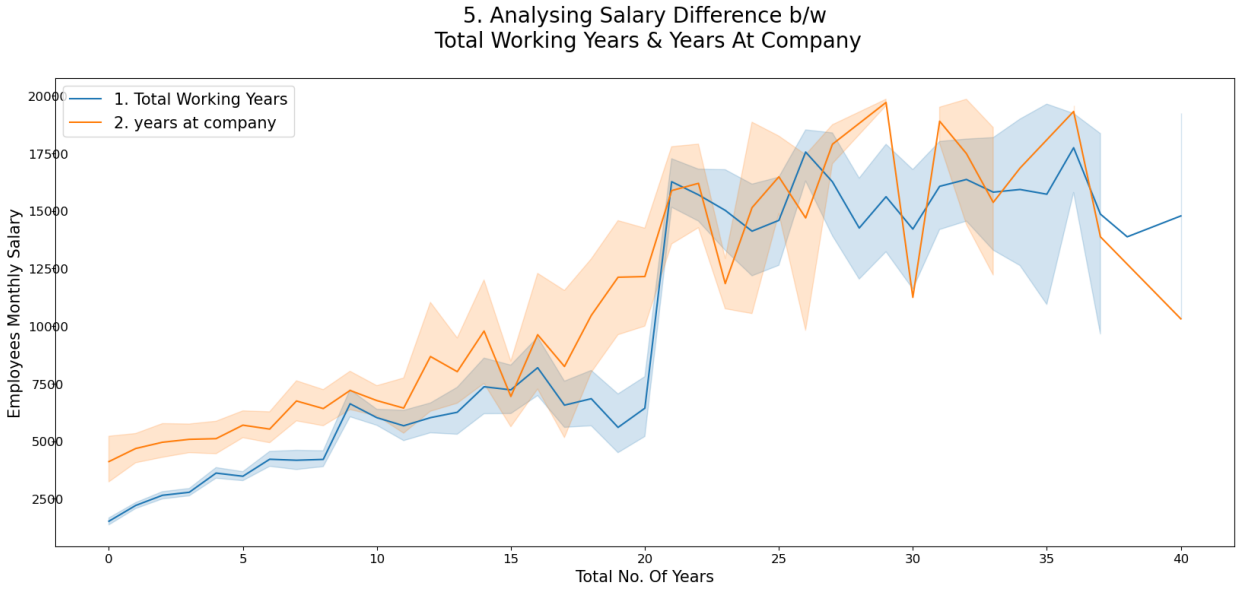
```
<Figure size 600x300 with 0 Axes>
```



In [ ]:

In [262]:
```
# 5) Analysing Salary Difference Between Total Working Years & Years At Company =====>>>>
```

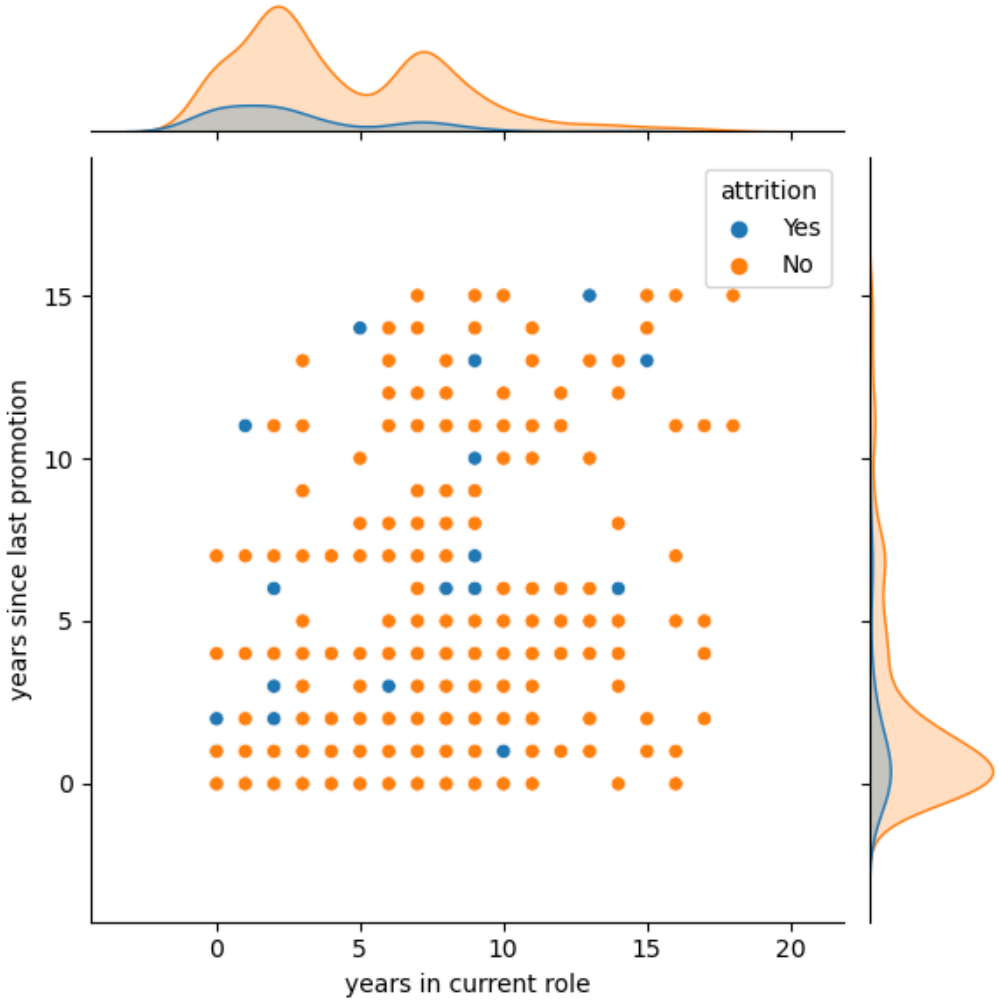
In [263]:
```
plt.figure(figsize=(20,8))
plt.title('\n 5. Analysing Salary Difference b/w \n Total Working Years & Years At Company\n',
sns.lineplot (data=df, x='total working years', y='monthly income', label = '1. Total Working
sns.lineplot (data=df, x='years at company', y='monthly income', label = '2. years at company'
# sns.lineplot (data=df, x='years in current role', y='monthly income', label = '3. job level',
# sns.lineplot (data=df, x='job role', y='monthly income', label = '4. job role')
plt.xlabel('Total No. Of Years',fontsize=15)
plt.xticks(rotation=0,ha='center',fontsize=12)
plt.ylabel('\nEmployees Monthly Salary',fontsize=15)
plt.yticks(rotation = 0, ha='center',fontsize=12)
plt.legend(loc='upper left', fontsize=15)
plt.show()

# From the below Analysis of Salary Difference between employees with Total Working Years & Yea
# ...there is a difference in salary upto 0-20 years, but after 20 years the salary of both are
# conclusion- company gives higher appreasels to their own earlier employees as compared to new
```



In [ ]:

In [264]:
```
# 6) Analysing Monthly salary with
```

In [265]:
```
sns.jointplot(data=df, x="years in current role",y='years since last promotion', hue="attritio
plt.show()

# here in the below we can find that the Atrrition Rate with the "Years in current role " & "ye
# ...is higher in between 0-5 years
```



In [ ]:
```

```

========UPTO HERE UNIVARIATE / BIVARIATE & MULTIVARIATE ANALYSIS IS COMPLETED
==========

In [ ]:
```

```

======= REMOVING / DROPPING NON-RELEVENT COLUMNS ==========

In [266]:
```
df.columns
```

Out[266]:
```
Index(['age', 'attrition', 'business travel', 'daily rate', 'department',
       'distance from home', 'education', 'education field', 'employeecount',
       'employee number', 'environment satisfaction', 'gender', 'hourly rate',
       'job involvement', 'job level', 'job role', 'job satisfaction',
       'marital status', 'monthly income', 'monthly rate',
       'num companies worked', 'over18', 'over time', 'percent salary hike',
       'performance rating', 'relationship satisfaction', 'standard hours',
       'stock option Level', 'total working years', 'training times last year',
       'work life balance', 'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

In [ ]:
```

```

In [267]:
```
# as according to analysis we found some following irrelevent columns in the given dataset,
# so we can remove the following columns from our dataset, which should not affect our result
# 1) employee count = it contains only '1' which is the only unique value in whole column, whi
# 2) employee number = it is just like an serial number assigned, non-relevent
# 3) over18 = offcourse all employees are above 18 years, so it is not MANDITORY to mention in
# 4) standard Hours = 80 hours is a standard hour for company, which only the unique value pres

# so these above 4 columns are NON-RELEVENT COLUMN'S of our dataset so we have to remove those
```

In [268]:
```
df.shape
# before removing column's , the shape of the dataset is 1470 rows & 35 columns.
```

Out[268]:
```
(1470, 35)
```

In [269]:
```
df.drop(['employeecount'],axis=1,inplace=True)
df.drop(['employee number'],axis=1,inplace=True)
df.drop(['over18'],axis=1,inplace=True)
df.drop(['standard hours'],axis=1,inplace=True)
```

In [270]:
```
df.shape
# here we can see the difference , 4 columns are successfully removed from our data set.
```

Out[270]:
```
(1470, 31)
```

In [ ]:
```

```

========== ENCODING DATASET ==========

In [271]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 31 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   age                        1470 non-null   int64 
 1   attrition                  1470 non-null   object
 2   business travel            1470 non-null   object
 3   daily rate                 1470 non-null   int64 
 4   department                 1470 non-null   object
 5   distance from home         1470 non-null   int64 
 6   education                  1470 non-null   int64 
 7   education field            1470 non-null   object
 8   environment satisfaction   1470 non-null   int64 
 9   gender                     1470 non-null   object
 10  hourly rate                1470 non-null   int64 
 11  job involvement            1470 non-null   int64 
 12  job level                  1470 non-null   int64 
 13  job role                   1470 non-null   object
 14  job satisfaction           1470 non-null   int64 
 15  marital status             1470 non-null   object
 16  monthly income             1470 non-null   int64 
 17  monthly rate               1470 non-null   int64 
 18  num companies worked       1470 non-null   int64 
 19  over time                  1470 non-null   object
 20  percent salary hike        1470 non-null   int64 
 21  performance rating         1470 non-null   int64 
 22  relationship satisfaction  1470 non-null   int64 
 23  stock option Level         1470 non-null   int64 
 24  total working years        1470 non-null   int64 
 25  training times last year   1470 non-null   int64 
 26  work life balance          1470 non-null   int64 
 27  years at company           1470 non-null   int64 
 28  years in current role      1470 non-null   int64 
 29  years since last promotion 1470 non-null   int64 
 30  years with currmanager     1470 non-null   int64 
dtypes: int64(23), object(8)
memory usage: 356.1+ KB
```

In [272]:
```
# here in our dataset we are having two 'object columns (8)'  which need's to be ENCODED for fu
# so here we are apllying "LABEL ENCODER" for all (8) columns:-
#  for which we have to import some libraries
```

In [273]:
```
from sklearn.preprocessing import LabelEncoder
```

In [274]:
```
le = LabelEncoder()
```

In [275]:
```
df["attrition"] = le.fit_transform(df["attrition"])
df["business travel"] = le.fit_transform(df["business travel"])
df["department"] = le.fit_transform(df["department"])
df["education field"] = le.fit_transform(df["education field"])
df["gender"] = le.fit_transform(df["gender"])
df["job role"] = le.fit_transform(df["job role"])
df["marital status"] = le.fit_transform(df["marital status"])
df["over time"] = le.fit_transform(df["over time"])
```

In [276]:
```
df.dtypes
# here below we can find that all OBJECT COLUMNS are converted successfully into INTEGER COLUM
```

Out[276]:
```
age                          int64
attrition                    int32
business travel              int32
daily rate                   int64
department                   int32
distance from home           int64
education                    int64
education field              int32
environment satisfaction     int64
gender                       int32
hourly rate                  int64
job involvement              int64
job level                    int64
job role                     int32
job satisfaction             int64
marital status               int32
monthly income               int64
monthly rate                 int64
num companies worked         int64
over time                    int32
percent salary hike          int64
performance rating           int64
relationship satisfaction    int64
stock option Level           int64
total working years          int64
training times last year     int64
work life balance            int64
years at company             int64
years in current role        int64
years since last promotion   int64
years with currmanager       int64
dtype: object
```

In [ ]:
```

```

========== FINDING CORRELATION IN DATASET ==========

In [277]:
```
cor = df.corr()
cor

# here finding non graphically correlation, here we can see that it is difficult to understand
# ....so further we find the correlation graphically by HEAT MAP.
```

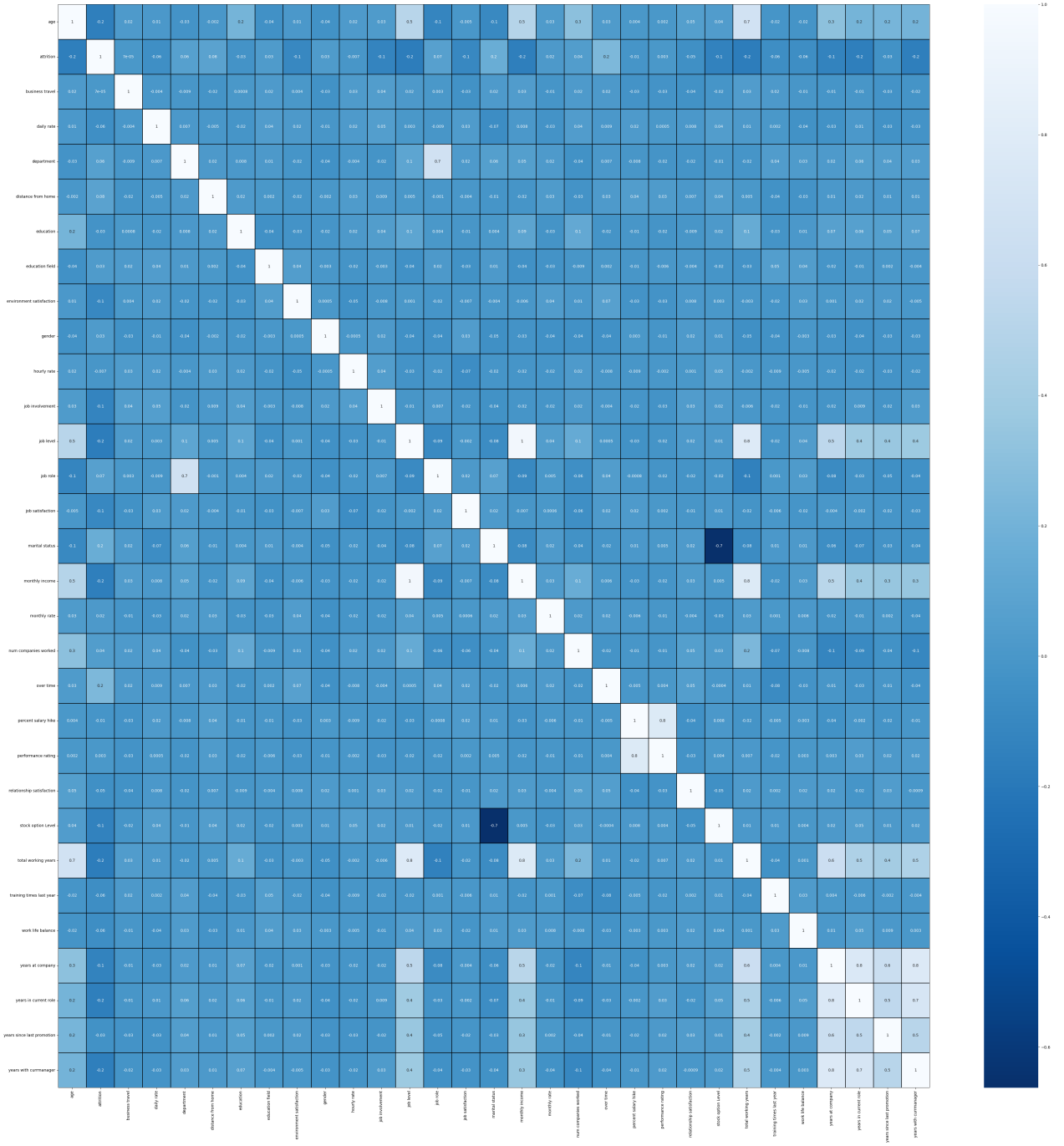
Out[277]:

	age	attrition	business travel	daily rate	department	distance from home	education	education field	environment satisfaction	
age	1.000000	-0.159205	0.024751	0.010661	-0.031882	-0.001686	0.208034	-0.040873	0.010146	-
attrition	-0.159205	1.000000	0.000074	-0.056652	0.063991	0.077924	-0.031373	0.026846	-0.103369	(
business travel	0.024751	0.000074	1.000000	-0.004086	-0.009044	-0.024469	0.000757	0.023724	0.004174	-(
daily rate	0.010661	-0.056652	-0.004086	1.000000	0.007109	-0.004985	-0.016806	0.037709	0.018355	-
department	-0.031882	0.063991	-0.009044	0.007109	1.000000	0.017225	0.007996	0.013720	-0.019395	-(
distance from home	-0.001686	0.077924	-0.024469	-0.004985	0.017225	1.000000	0.021042	0.002013	-0.016075	-(
education	0.208034	-0.031373	0.000757	-0.016806	0.007996	0.021042	1.000000	-0.039592	-0.027128	-(
education field	-0.040873	0.026846	0.023724	0.037709	0.013720	0.002013	-0.039592	1.000000	0.043163	-(
environment satisfaction	0.010146	-0.103369	0.004174	0.018355	-0.019395	-0.016075	-0.027128	0.043163	1.000000	(
gender	-0.036311	0.029453	-0.032981	-0.011716	-0.041583	-0.001851	-0.016547	-0.002504	0.000508	
hourly rate	0.024287	-0.006846	0.026528	0.023381	-0.004144	0.031131	0.016775	-0.021941	-0.049857	-(
job involvement	0.029820	-0.130016	0.039062	0.046135	-0.024586	0.008783	0.042438	-0.002655	-0.008278	(
job level	0.509604	-0.169105	0.019311	0.002966	0.101963	0.005303	0.101589	-0.044933	0.001212	-(
job role	-0.122427	0.067151	0.002724	-0.009472	0.662431	-0.001015	0.004236	0.015599	-0.017321	-(
job satisfaction	-0.004892	-0.103481	-0.033962	0.030571	0.021001	-0.003669	-0.011296	-0.034401	-0.006784	(
marital status	-0.095029	0.162070	0.024001	-0.069586	0.056073	-0.014437	0.004053	0.014420	-0.003593	-(
monthly income	0.497855	-0.159840	0.034319	0.007707	0.053130	-0.017014	0.094961	-0.041070	-0.006259	-(
monthly rate	0.028051	0.015170	-0.014107	-0.032182	0.023642	0.027473	-0.026084	-0.027182	0.037600	-(
num companies worked	0.299635	0.043494	0.020875	0.038153	-0.035882	-0.029251	0.126317	-0.008663	0.012594	-(
over time	0.028062	0.246118	0.016543	0.009135	0.007481	0.025514	-0.020322	0.002259	0.070132	-(
percent salary hike	0.003634	-0.013478	-0.029377	0.022704	-0.007840	0.040235	-0.011111	-0.011214	-0.031701	(
performance rating	0.001904	0.002889	-0.026341	0.000473	-0.024604	0.027110	-0.024539	-0.005614	-0.029548	-(
relationship satisfaction	0.053535	-0.045872	-0.035986	0.007846	-0.022414	0.006557	-0.009118	-0.004378	0.007665	(
stock option Level	0.037510	-0.137145	-0.016727	0.042143	-0.012193	0.044872	0.018422	-0.016185	0.003432	(
total working years	0.680381	-0.171063	0.034226	0.014515	-0.015762	0.004628	0.148280	-0.027848	-0.002693	-(
training times last year	-0.019621	-0.059478	0.015240	0.002453	0.036875	-0.036942	-0.025100	0.049195	-0.019359	-(
work life balance	-0.021490	-0.063939	-0.011256	-0.037848	0.026383	-0.026556	0.009819	0.041191	0.027627	-(
years at company	0.311309	-0.134392	-0.014575	-0.034055	0.022920	0.009508	0.069114	-0.018692	0.001458	-(

	age	attrition	business travel	daily rate	department	distance from home	education	education field	environment satisfaction	
years in current role	0.212901	-0.160545	-0.011497	0.009932	0.056315	0.018845	0.060236	-0.010506	0.018007	-
years since last promotion	0.216513	-0.033019	-0.032591	-0.033229	0.040061	0.010029	0.054254	0.002326	0.016194	-
years with currmanager	0.202089	-0.156199	-0.022636	-0.026363	0.034282	0.014406	0.069065	-0.004130	-0.004999	-

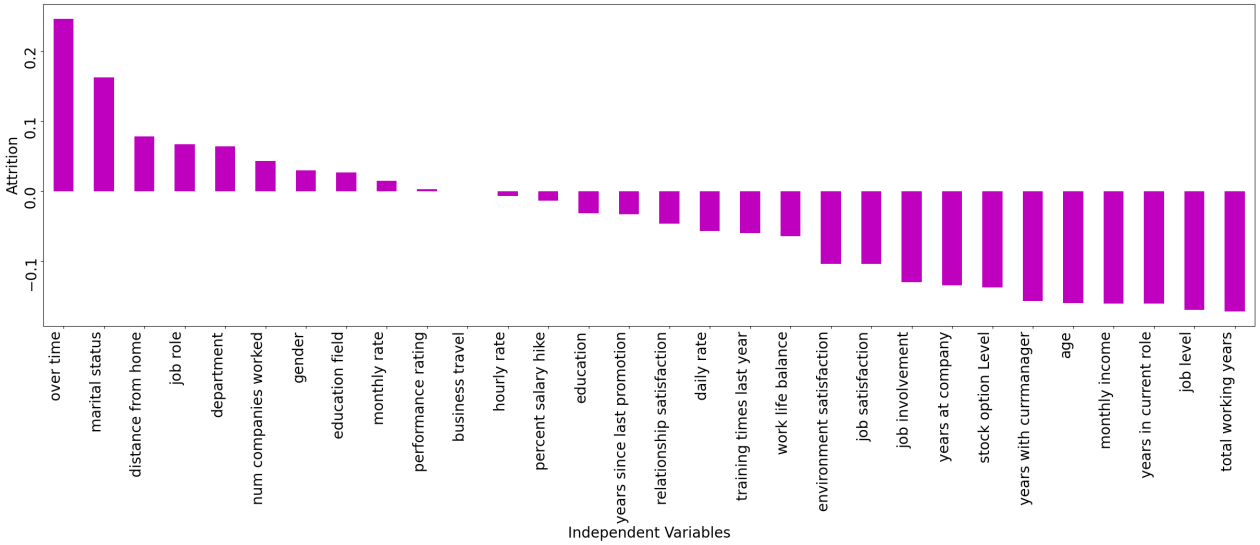
31 rows × 31 columns

In [278]:
```
plt.figure (figsize = (50,50), facecolor = "white")
sns.heatmap(df.corr(),linewidth=0.1,fmt="0.1g",linecolor="black",annot=True,cmap="Blues_r")
plt.yticks(rotation=0);
plt.show()


# here below due to large no. of columms it is difficult to find correlation.
# so we can use another method to make it clear.
```



In [ ]:
```
```

In [279]:
```
plt.figure(figsize=(30,8))
df.corr()['attrition'].sort_values(ascending=False).drop(['attrition']).plot(kind='bar',color=
plt.xlabel('Independent Variables',fontsize=20)
plt.xticks(rotation=90,ha='right',fontsize=20)
plt.ylabel('Attrition',fontsize =20)
plt.yticks(rotation=90,ha='right',fontsize=20)
plt.title=("Correlation with Attrition")
plt.show()

# here in below graph we can clearly see that NO. OF CORRELATED COLUMNS are LESS ,as compared t
# The Highest Correlated Column is = OVER TIME
#     2nd Highest Correlated Column is = MARITAL STATUS
# The NEUTRAL COLUMN with TARGET COLUMN is = =BUSINESS TRAVEL
# The Highest Negative Correlated Column is = TOTAL WORKING YEARS < JOB LEVEL
```



In [280]:
```
cor['attrition'].sort_values(ascending=False)

# the same above results in tavel formate.
#  here we can see with 'attrition' the highest correlated column is - 'overtime'
#  but here we can see the 'value of correlation', i.e  0.24  for overtime , which is highly co
#  it means that the highly correlated column is not having highly correlation with the target
# ...compared to other columns.
```

Out[280]:
```
attrition                    1.000000
over time                    0.246118
marital status               0.162070
distance from home           0.077924
job role                     0.067151
department                   0.063991
num companies worked         0.043494
gender                       0.029453
education field              0.026846
monthly rate                 0.015170
performance rating           0.002889
business travel              0.000074
hourly rate                 -0.006846
percent salary hike         -0.013478
education                   -0.031373
years since last promotion  -0.033019
relationship satisfaction   -0.045872
daily rate                  -0.056652
training times last year    -0.059478
work life balance           -0.063939
environment satisfaction    -0.103369
job satisfaction            -0.103481
job involvement             -0.130016
years at company            -0.134392
stock option Level          -0.137145
years with currmanager      -0.156199
age                         -0.159205
monthly income              -0.159840
years in current role       -0.160545
job level                   -0.169105
total working years         -0.171063
Name: attrition, dtype: float64
```

In [ ]:
```

```

CHECKING FOR OUTLIERS

==========

In [ ]:
```

```

In [281]:
```
df.describe()
# here in the describe methode we are getting soo many STATISTICAL INFORMATION about the datase
# 1. first of all above we are getting 'count' for each of the column.
#                 as we know the total number of row counts for each column is 1,470. and here
#                 ... column is smame. not a single blank/'nan' is present in any of the columr

# 2. MEAN : In this, we can get MEAN VALUE for the every column.
# 3. STD : which is Standard Deviation , which shows that how the data of the column is deviate
# 4. MIN : It shows the Minimum value present in the column.
# 5. 25% : It gives us the 25th Percentile Value in the column.
# 6. 50% : It gives us the 50th Percentile Value in the column.
# 7. 75% : It gives us the 75th Percentile Value in the column.
# 8. Max : It gives us the MAXIMUM VALUE present the column.

# As If in any column the Difference between the value at 75th Percentile & MAX is Higher then,
# so we have to check the 75th% & MAX for each of the column.
#  here we find that in the following columns there is huge difference betweeen 75% & MAX :
#  1)- daily rate, 2)- distance from home, 3)-Total working years, 4)- years at company, 5)- ye
# ....6)- years since last promotion, 7)- years with current Manager
# so in the above mentioned columns there may be presence of outliers, but we have to check al
```

Out[281]:

	age	attrition	business travel	daily rate	department	distance from home	education	education field	env sa
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	147
mean	36.923810	0.161224	1.607483	802.485714	1.260544	9.192517	2.912925	2.247619	
std	9.135373	0.367863	0.665455	403.509100	0.527792	8.106864	1.024165	1.331369	
min	18.000000	0.000000	0.000000	102.000000	0.000000	1.000000	1.000000	0.000000	
25%	30.000000	0.000000	1.000000	465.000000	1.000000	2.000000	2.000000	1.000000	
50%	36.000000	0.000000	2.000000	802.000000	1.000000	7.000000	3.000000	2.000000	
75%	43.000000	0.000000	2.000000	1157.000000	2.000000	14.000000	4.000000	3.000000	
max	60.000000	1.000000	2.000000	1499.000000	2.000000	29.000000	5.000000	5.000000	

8 rows × 31 columns

In [282]:
```
# By using BOXPLOT METHOD we can chek PRESENCE OF OUTLIERS  in every column.
```

In [283]:
```
df.columns
```

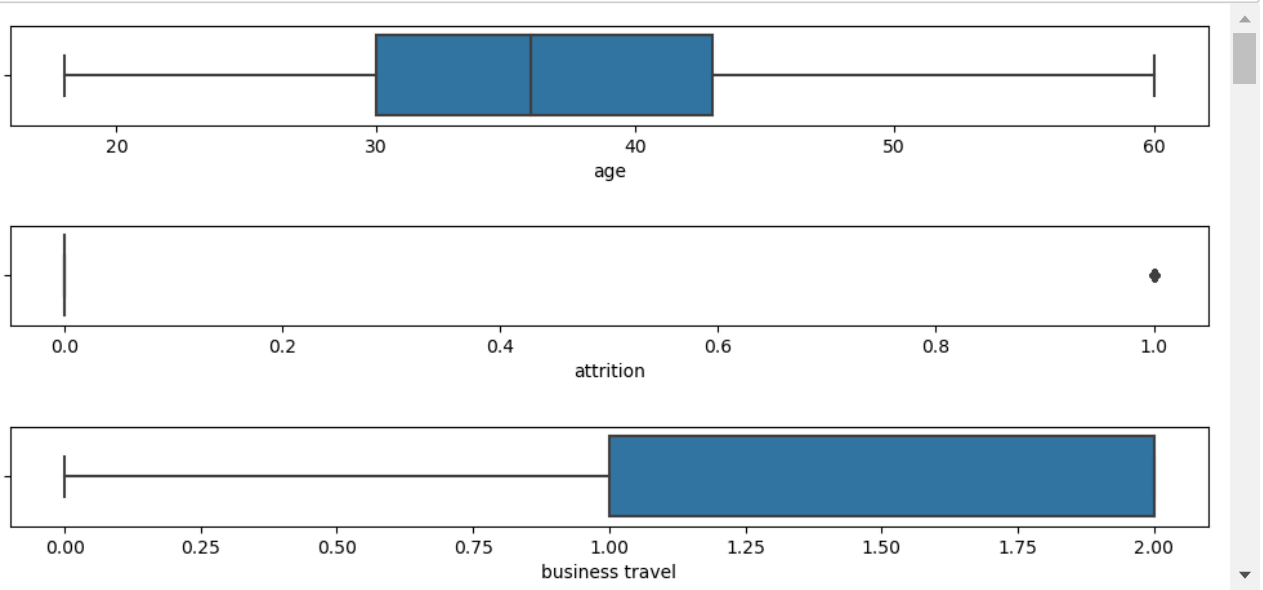
Out[283]:
```
Index(['age', 'attrition', 'business travel', 'daily rate', 'department',
       'distance from home', 'education', 'education field',
       'environment satisfaction', 'gender', 'hourly rate', 'job involvement',
       'job level', 'job role', 'job satisfaction', 'marital status',
       'monthly income', 'monthly rate', 'num companies worked', 'over time',
       'percent salary hike', 'performance rating',
       'relationship satisfaction', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

In [284]:
```
df.columns.nunique()
```

Out[284]:
```
31
```

In [285]:
```
#  by using forloop we can chek outliers for each column in a single code.
```

In [286]:
```
for i in df.columns[0:31]:
    plt.figure (figsize = (12,1), facecolor = "white")
    sns.boxplot(x=i,data=df)
    plt.show()

#  here below we can find the outliers for all the cloumns by using boxplot.
#  and we are found outliers in :
#  (1)Monthly income, (2) num companies worked,(3) stock option level,(4) total working years,
# ..(6) years at company, (7) years in current role,(8) years since last promotion,(9) years w
# .....are having outliers.
# so out of 31 columns we found OUTLIERS IN 9 COLUMNS , now we have to remove those outliers fr
```



In [287]:
```
#  (1)Monthly income, (2) num companies worked,(3) stock option level,(4) total working years,
# ..(6) years at company, (7) years in current role,(8) years since last promotion,(9) years w
```

========== REMOVING OF OUTLIERS BY USING Z-SCORE METHOD ==========

In [288]:
```
#  we can not remove outliers from out TARGET COLUMN, so first we have to seprate target columr
# For this first we need to identify the ZSCORE VALUES, for which we have to import some librar
```

In [289]:
```
from scipy.stats import zscore
```

In [290]:
```
z = np.abs(zscore(df))
z.head(5)

# by applying 'abs' (absolute method), we are getting all the entries whose z-score value is po
#  Ideally we can call the OUTLIERS whos ZSCORE VALUE is LESS THEN 3 AND MORE THEN 3
# so we have to remove all the data whose ZSCORE >3 & <3
#  below here we apllying "abs" i.e absolute method it returns us the all zscore values greater
#  so we just need to remove lesserr then 3 zscore values.
```

Out[290]:

	age	attrition	business travel	daily rate	department	distance from home	education	education field	environment satisfaction	gender	...	p
0	0.446350	2.280906	0.590048	0.742527	1.401512	1.010909	0.891688	0.937414	0.660531	1.224745	...	
1	1.322365	0.438422	0.913194	1.297775	0.493817	0.147150	1.868426	0.937414	0.254625	0.816497	...	
2	0.008343	2.280906	0.590048	1.414363	0.493817	0.887515	0.891688	1.316673	1.169781	0.816497	...	
3	0.429664	0.438422	0.913194	1.461466	0.493817	0.764121	1.061787	0.937414	1.169781	1.224745	...	
4	1.086676	0.438422	0.590048	0.524295	0.493817	0.887515	1.868426	0.565311	1.575686	0.816497	...	

5 rows × 31 columns

In [291]:
```
threshold = 3
print(np.where(z>3))

# here below we found only 110 values,  whose z-score is more then > 3
# i.e means we are having 110 values which might be outliers,are still present in our dataset,
# ...and we have to remove those outliers
```

```
(array([  28,   45,   62,   62,   63,   64,   85,   98,   98,  110,  123,
        123,  123,  126,  126,  126,  153,  178,  187,  187,  190,  190,
        218,  231,  231,  237,  237,  270,  270,  281,  326,  386,  386,
        401,  411,  425,  425,  427,  445,  466,  473,  477,  535,  561,
        561,  584,  592,  595,  595,  595,  616,  624,  635,  653,  653,
        677,  686,  701,  716,  746,  749,  752,  799,  838,  861,  861,
        875,  875,  894,  914,  914,  918,  922,  926,  926,  937,  956,
        962,  976,  976, 1008, 1024, 1043, 1078, 1078, 1086, 1086, 1093,
       1111, 1116, 1116, 1135, 1138, 1138, 1156, 1184, 1221, 1223, 1242,
       1295, 1301, 1301, 1303, 1327, 1331, 1348, 1351, 1401, 1414, 1430],
      dtype=int64), array([30, 29, 27, 29, 28, 29, 24, 24, 27, 29, 28, 29, 30, 24, 27, 29, 3
0,
       29, 24, 30, 27, 28, 29, 28, 30, 27, 29, 24, 27, 28, 29, 29, 30, 24,
       27, 27, 29, 29, 24, 28, 27, 27, 29, 27, 30, 29, 27, 24, 27, 29, 30,
       24, 30, 27, 29, 27, 30, 29, 28, 28, 27, 29, 29, 29, 27, 29, 29, 30,
       24, 27, 29, 27, 29, 29, 30, 29, 24, 27, 28, 29, 29, 28, 24, 29, 30,
       27, 29, 29, 27, 24, 27, 27, 27, 29, 29, 24, 29, 29, 29, 29, 24, 29,
       29, 28, 29, 30, 28, 24, 29, 28], dtype=int64))
```

In [292]:
```
df_new = df[(z<3).all(axis=1)]
df_new

# here we can see the difference clearly that, earlier there was 1470 total rows are there, and
# ...there are only 1387 rows are present in our dataset.
# so there are  83 OUTLIERS are removed during this process.
```

Out[292]:

	age	attrition	business travel	daily rate	department	distance from home	education	education field	environment satisfaction	gender	...	performar rat
0	41	1	2	1102	2	1	2	1	2	0	...	
1	49	0	1	279	1	8	1	1	3	1	...	
2	37	1	2	1373	1	2	2	4	4	1	...	
3	33	0	1	1392	1	3	4	1	4	0	...	
4	27	0	2	591	1	2	1	3	1	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	0	1	884	1	23	2	3	3	1	...	
1466	39	0	2	613	1	6	1	3	4	1	...	
1467	27	0	2	155	1	4	3	1	2	1	...	
1468	49	0	1	1023	2	2	3	3	4	1	...	
1469	34	0	2	628	1	8	3	3	2	1	...	

1387 rows × 31 columns

In [293]:
```
df.shape
```

Out[293]: (1470, 31)

In [294]:
```
df_new.shape
```

Out[294]: (1387, 31)

In [295]:
```
#  here above we can clearly see the difference, 83 outliers are removed from the new dataset.
```

==========CHECKING REMOVAL OF OUTLIERS BY BOXPLOT (COMPARING 'df' & 'df_new')
==========

In [ ]:
```
```

In [296]:
```
df_new.columns
```

Out[296]:
```
Index(['age', 'attrition', 'business travel', 'daily rate', 'department',
       'distance from home', 'education', 'education field',
       'environment satisfaction', 'gender', 'hourly rate', 'job involvement',
       'job level', 'job role', 'job satisfaction', 'marital status',
       'monthly income', 'monthly rate', 'num companies worked', 'over time',
       'percent salary hike', 'performance rating',
       'relationship satisfaction', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

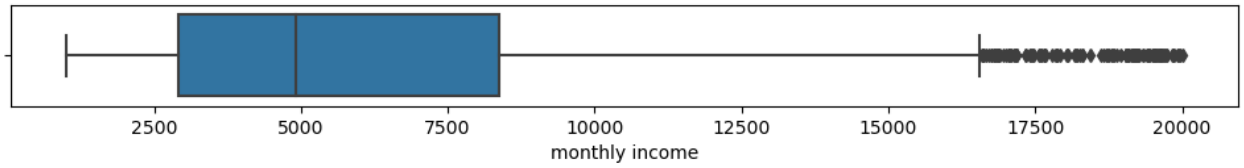
In [297]:
```
df_new.columns.nunique()
```

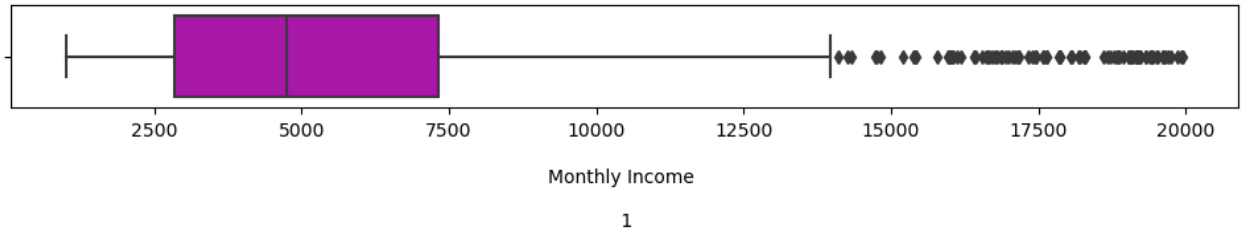
Out[297]: 31

In [298]:
```
# now in the earlier findings we found that there are may presence of outliers in some of the c
# ....i.e (#  (1)Monthly income, (2) num companies worked,(3) stock option level,(4) total work
# ..(6) years at company, (7) years in current role,(8) years since last promotion,(9) years wi
#  now we have to check out of these columns which outlier hasbeen removed from the dataset.
```

In [299]:
```
# 1) Analysing MONTHLY INCOME after removing OUTLIERS ====>>>
```

In [300]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='monthly income',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='monthly income',data=df_new, color='m')
plt.xlabel('\nMonthly Income \n\n 1')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'MONTHLY INCOME'
```

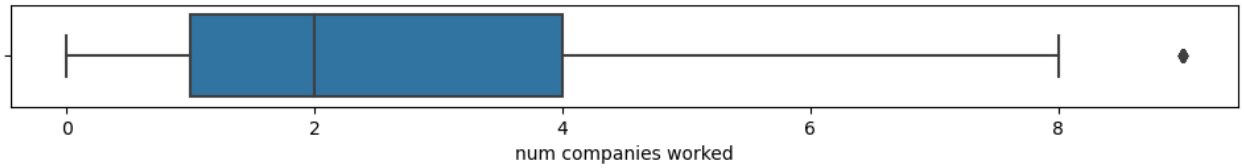


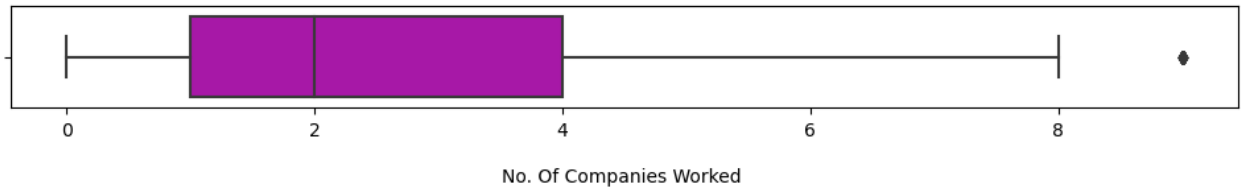


In [ ]:

In [301]:
```
# 2) Analysing Removal of Outliers in 'Num companies Worked'=====>>>
```

In [302]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='num companies worked',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='num companies worked',data=df_new, color='m')
plt.xlabel('\nNo. Of Companies Worked \n\n 2')
plt.show()

# NOT CONSIDERED AS A OUTLIER BY ALGORITHM
```



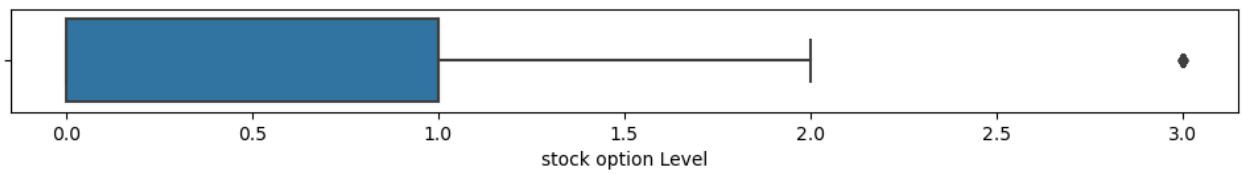


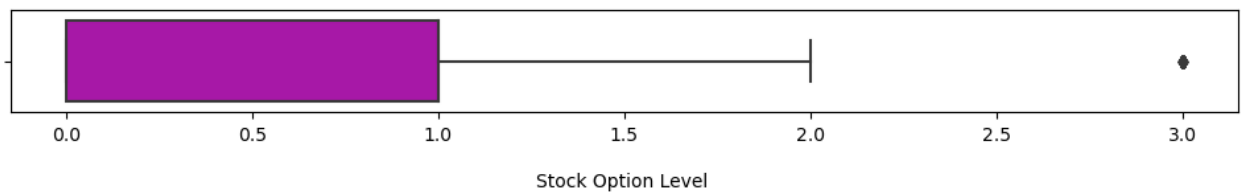
2

In [ ]:
```

```

In [303]:
```
# 3) Analysing Outliers in 'Stock Option Level'======>>>
```

In [304]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='stock option Level',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='stock option Level',data=df_new, color='m')
plt.xlabel('\nStock Option Level \n\n 3')
plt.show()

# here also algorithm doest not consider this as a OUTLIER
```



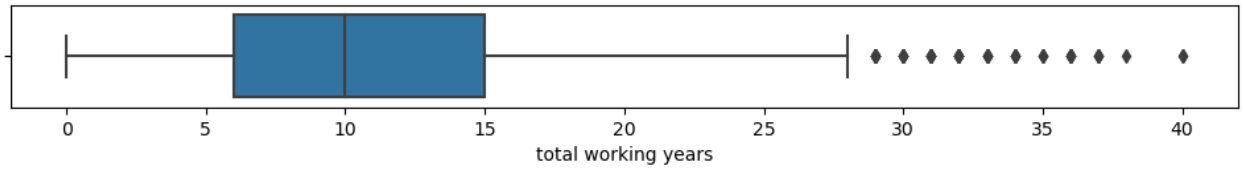


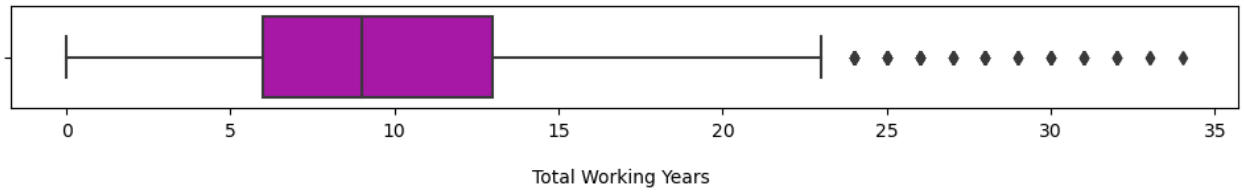
3

In [ ]:

In [305]:
```
# 4) Analysing OUTLIERS in 'Total Working Years' ======>>>>>
```

In [306]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='total working years',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='total working years',data=df_new, color='m')
plt.xlabel('\nTotal Working Years \n\n 4')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'TOTAL WORKING YE
```



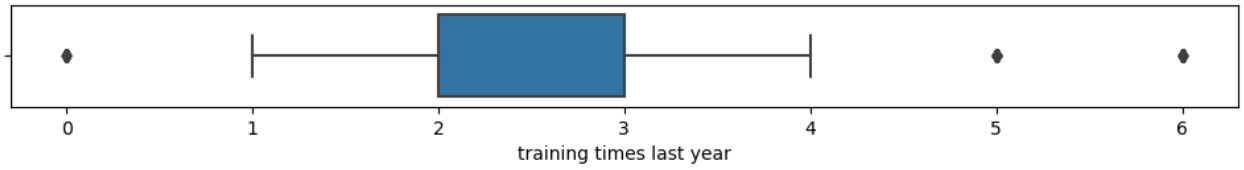


4

In [ ]:

In [307]:
```
# 5) Analysing OUTLIERS in Training Time Last Year =====>>>>
```

In [308]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='training times last year',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='training times last year',data=df_new, color='m')
plt.xlabel('\nTraining Time Last Year \n\n 5')
plt.show()

# here algorithm doesnot consider this as a OUTLIERS, therfore it can't be removed
```

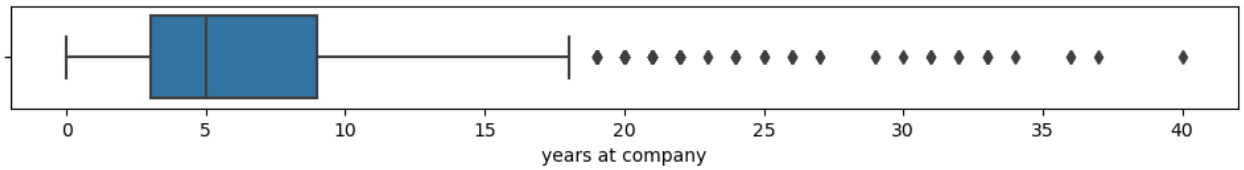


5

In [ ]:
```
```

In [309]:
```
# 6) Analysign Outliers in
```

In [310]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years at company',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years at company',data=df_new, color='m')
plt.xlabel('\nYears At Company \n\n 6')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'YEARS AT COMPANY
```

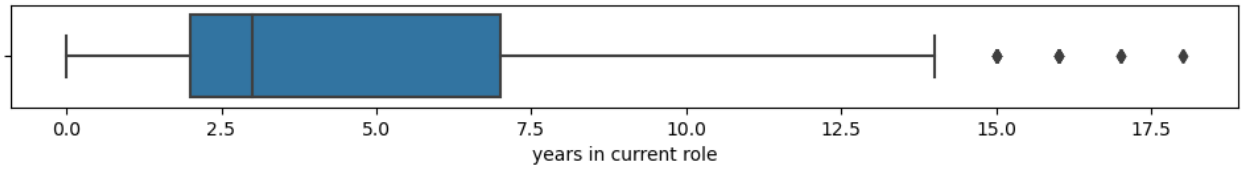


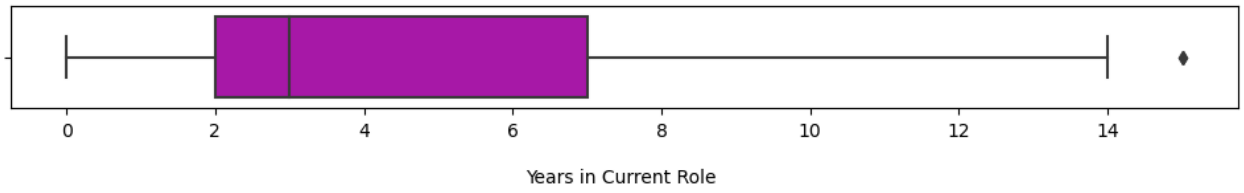
6

In [ ]:

In [311]:
```
# 7) Analysing Outliers in YEARS AT CURRENT ROLE ====>>>>
```

In [312]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years in current role',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years in current role',data=df_new, color='m')
plt.xlabel('\nYears in Current Role \n\n 7')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'YEARS IN CURRENT
```



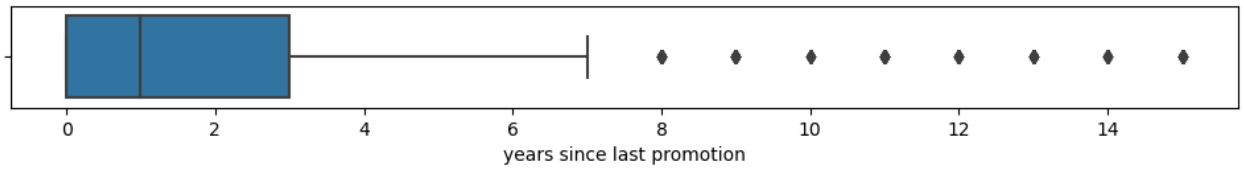


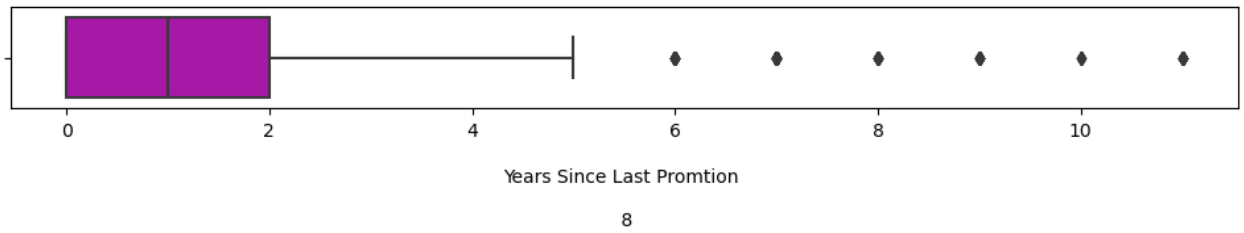
7

In [ ]:

In [313]:
```
# 8) Analysing Outliers in Years Since Last Promotion column =====>>>>>
```

In [314]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years since last promotion',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years since last promotion',data=df_new, color='m')
plt.xlabel('\nYears Since Last Promtion \n\n 8')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'YEARS SINCE LAS
```



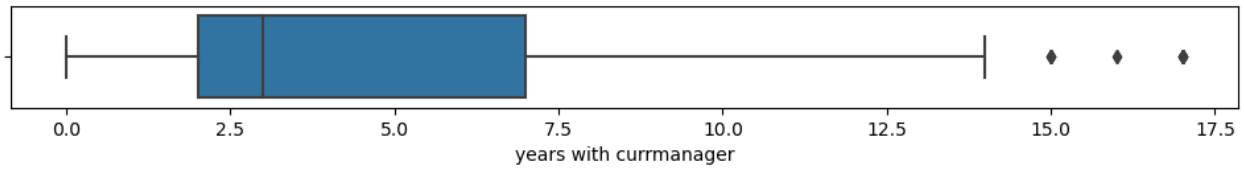


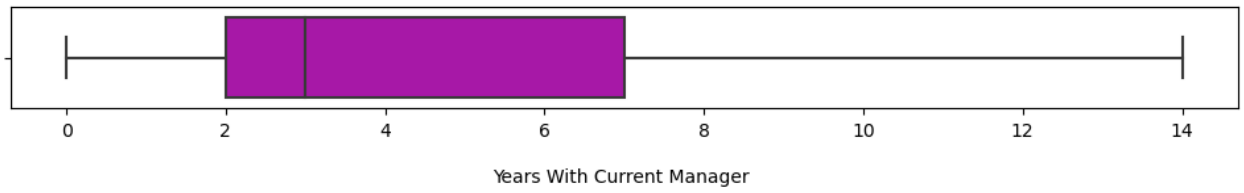
8

In [ ]:

In [315]:
```
# 9) Analysing Outliers in Years With Current Manager =====>>>>
```

In [316]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years with currmanager',data=df)
plt.show()

# it is the EARLIER (df dataset) WITH PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='years with currmanager',data=df_new, color='m')
plt.xlabel('\nYears With Current Manager \n\n 9')
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , outlier which is removed above by Z-SCORE METHOD is from 'YEARS WITH CURRE
```





9

In [ ]:
```

```

========== FINDING & REMOVAL OF OUTLIERS ARE COMPLETED ==========

In [ ]:
```

```

CHECKING SKEWNESS ==========

In [317]:
```
# the skewness shows the distribution of data, if the data is widely skewed that means it is no
#  ideal range of skewness is ( -0.5 to +0.5)
# We can't remove skewness from our Target Column
```

In [318]:
```
df_new.skew()
# here in the below table we can see the skewness in the following columns:
# attrition, job level, monthly income, num companies worked, performance rating, total working
#  years since last promotion
#  so we have to remove the skewness from the mentioned columns for better results.
```

Out[318]:
```
age                          0.472280
attrition                    1.805983
business travel             -1.426774
daily rate                  -0.017078
department                   0.183919
distance from home           0.954752
education                   -0.289024
education field              0.544868
environment satisfaction    -0.325285
gender                      -0.417296
hourly rate                 -0.030481
job involvement             -0.501401
job level                    1.126075
job role                    -0.386843
job satisfaction            -0.345612
marital status              -0.160952
monthly income               1.544770
monthly rate                 0.030596
num companies worked         1.037715
over time                    0.954751
percent salary hike          0.800592
performance rating           1.931566
relationship satisfaction   -0.295686
stock option Level           0.962332
total working years          1.034487
training times last year     0.577614
work life balance           -0.557100
years at company             1.248623
years in current role        0.726675
years since last promotion   1.756335
years with currmanager       0.694506
dtype: float64
```

In [319]:
```
# so we have to remove skewness from those columns by using 'cuberoot' method.
```

In [320]:
```
# attrition, job level, monthly income, num companies worked, performance rating, total working
#  years since last promotion
```

In [321]:
```
df_new['job level'] = np.cbrt(df_new['job level'])
df_new['monthly income'] = np.cbrt(df_new['monthly income'])
df_new['num companies worked'] = np.cbrt(df_new['num companies worked'])
df_new['performance rating'] = np.cbrt(df_new['performance rating'])
df_new['total working years'] = np.cbrt(df_new['total working years'])
df_new['years at company'] = np.cbrt(df_new['years at company'])
df_new['years since last promotion'] = np.cbrt(df_new['years since last promotion'])
```

In [322]:
```
df_new.skew()
# here the maximum skewness is removed from the dataset.
```

Out[322]:
```
age                          0.472280
attrition                    1.805983
business travel             -1.426774
daily rate                  -0.017078
department                   0.183919
distance from home           0.954752
education                   -0.289024
education field              0.544868
environment satisfaction    -0.325285
gender                      -0.417296
hourly rate                 -0.030481
job involvement             -0.501401
job level                    0.508238
job role                    -0.386843
job satisfaction            -0.345612
marital status              -0.160952
monthly income               0.742985
monthly rate                 0.030596
num companies worked        -0.689329
over time                    0.954751
percent salary hike          0.800592
performance rating           1.931566
relationship satisfaction   -0.295686
stock option Level           0.962332
total working years         -0.563399
training times last year     0.577614
work life balance           -0.557100
years at company            -0.700051
years in current role        0.726675
years since last promotion   0.157801
years with currmanager       0.694506
dtype: float64
```

In [ ]:
```

```

========== REMOVED SKEWNESS SUCCESSFULLY
==========

In [323]:
```
cor['attrition'].sort_values(ascending=False)
# here above we can find that 'business travel' & 'Performance rating' both columns are very ne
# ....target column.
# so we can also drop them for better results...
```

Out[323]:
```
attrition                    1.000000
over time                    0.246118
marital status               0.162070
distance from home           0.077924
job role                     0.067151
department                   0.063991
num companies worked         0.043494
gender                       0.029453
education field              0.026846
monthly rate                 0.015170
performance rating           0.002889
business travel              0.000074
hourly rate                 -0.006846
percent salary hike         -0.013478
education                   -0.031373
years since last promotion  -0.033019
relationship satisfaction   -0.045872
daily rate                  -0.056652
training times last year    -0.059478
work life balance           -0.063939
environment satisfaction    -0.103369
job satisfaction            -0.103481
job involvement             -0.130016
years at company            -0.134392
stock option Level          -0.137145
years with currmanager      -0.156199
age                         -0.159205
monthly income              -0.159840
years in current role       -0.160545
job level                   -0.169105
total working years         -0.171063
Name: attrition, dtype: float64
```

In [324]:
```
df_new.drop(['business travel'],axis=1,inplace=True)
```

In [325]:
```
df.drop(['performance rating'],axis=1,inplace=True)
```

In [326]:
```
df_new.skew()
```

Out[326]:
```
age                          0.472280
attrition                    1.805983
daily rate                  -0.017078
department                   0.183919
distance from home           0.954752
education                   -0.289024
education field              0.544868
environment satisfaction    -0.325285
gender                      -0.417296
hourly rate                 -0.030481
job involvement             -0.501401
job level                    0.508238
job role                    -0.386843
job satisfaction            -0.345612
marital status              -0.160952
monthly income               0.742985
monthly rate                 0.030596
num companies worked        -0.689329
over time                    0.954751
percent salary hike          0.800592
performance rating           1.931566
relationship satisfaction   -0.295686
stock option Level           0.962332
total working years         -0.563399
training times last year     0.577614
work life balance           -0.557100
years at company            -0.700051
years in current role        0.726675
years since last promotion   0.157801
years with currmanager       0.694506
dtype: float64
```

In [327]:
```
df_new.head(2)
```

Out[327]:

	age	attrition	daily rate	department	distance from home	education	education field	environment satisfaction	gender	hourly rate	...	performance rating	r s
0	41	1	1102	2	1	2	1	2	0	94	...	1.442250	
1	49	0	279	1	8	1	1	3	1	61	...	1.587401	

2 rows × 30 columns

DIVIDING DATA INTO INDEPENDENT & TARGET VARIABLE

==========

In [328]:
```
df_new.shape
```

Out[328]:
```
(1387, 30)
```

In [329]:
```
df_new.columns
```

Out[329]:
```
Index(['age', 'attrition', 'daily rate', 'department', 'distance from home',
       'education', 'education field', 'environment satisfaction', 'gender',
       'hourly rate', 'job involvement', 'job level', 'job role',
       'job satisfaction', 'marital status', 'monthly income', 'monthly rate',
       'num companies worked', 'over time', 'percent salary hike',
       'performance rating', 'relationship satisfaction', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

In [330]:
```
x = df_new[['age', 'daily rate', 'department',
       'distance from home', 'education', 'education field',
       'environment satisfaction', 'gender', 'hourly rate', 'job involvement',
       'job level', 'job role', 'job satisfaction', 'marital status',
       'monthly income', 'monthly rate', 'num companies worked', 'over time',
       'percent salary hike',
       'relationship satisfaction', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager']]
# also removing 'performance rating' column
```

In [331]:
```
y=df_new[['attrition']]
```

In [332]:
```
x.head(2)
```

Out[332]:

	age	daily rate	department	distance from home	education	education field	environment satisfaction	gender	hourly rate	job involvement	...	percent salary hike	re s
0	41	1102	2	1	2	1	2	0	94	3	...	11	
1	49	279	1	8	1	1	3	1	61	2	...	23	

2 rows × 28 columns

In [333]:
```
y.head(2)
```

Out[333]:

	attrition
0	1
1	0

In [334]:
```
x.shape
```

Out[334]:
```
(1387, 28)
```

In [335]:
```
y.shape
```

Out[335]:
```
(1387, 1)
```

In [336]:
```
# here above we seprates independent and dependent columns successfully.
```

APPLYING SCALING TECHNIQUES

In [337]:
```
# here we need to apply scaling techniques on our dataset,by scaling techniques we normalise th
#  we can't apply SCALING TECHNIQUES on TARGET VARIABLE
#  to aplly scaling techinuque we need to import some libraries first.
```

In [338]:
```
from sklearn.preprocessing import StandardScaler
```

In [339]:
```
st = StandardScaler()
```

In [340]:
```
x = st.fit_transform(x)
x
```

Out[340]:
```
array([[ 0.5366811 ,  0.73432467,  1.40537338, ...,  0.04090327,
        -1.09250316,  0.36826963],
       [ 1.44211114, -1.30776866, -0.49633675, ...,  0.96589518,
         0.29458159,  0.9821999 ],
       [ 0.08396607,  1.40675151, -0.49633675, ..., -1.19241928,
        -1.09250316, -1.16655603],
       ...,
       [-1.04782149, -1.61544736, -0.49633675, ..., -0.575758  ,
        -1.09250316, -0.24566063],
       [ 1.44211114,  0.53830356,  1.40537338, ...,  0.65756454,
        -1.09250316,  1.28916503],
       [-0.2555702 , -0.44180199, -0.49633675, ..., -0.26742737,
         0.29458159, -0.55262577]])
```

In [341]:
```
xf = pd.DataFrame(data=x)
print(xf)

# here we get our dataset (xf1) after applying SCALING TECHING (STANDARD SCALER)
```

```
            0         1         2         3         4         5         6   \
0     0.536681  0.734325  1.405373 -1.011249 -0.876177 -0.940815 -0.665328
1     1.442111 -1.307769 -0.496337 -0.145521 -1.853858 -0.940815  0.251978
2     0.083966  1.406752 -0.496337 -0.887573 -0.876177  1.305159  1.169285
3    -0.368749  1.453896 -0.496337 -0.763898  1.079185 -0.940815  1.169285
4    -1.047821 -0.533609 -0.496337 -0.887573 -1.853858  0.556501 -1.582635
...        ...       ...       ...       ...       ...       ...       ...
1382 -0.029213  0.193406 -0.496337  1.709609 -0.876177  0.556501  0.251978
1383  0.310324 -0.479021 -0.496337 -0.392872 -1.853858  0.556501  1.169285
1384 -1.047821 -1.615447 -0.496337 -0.640223  0.101504 -0.940815 -0.665328
1385  1.442111  0.538304  1.405373 -0.887573  0.101504  0.556501  1.169285
1386 -0.255570 -0.441802 -0.496337 -0.145521  0.101504  0.556501 -0.665328

            7         8         9   ...        18        19        20  \
0    -1.229911  1.388670  0.376231  ... -1.161414 -1.575817 -0.929427
1     0.813067 -0.239091 -1.034126  ...  2.146916  1.199034  0.238250
2     0.813067  1.290017 -1.034126  ... -0.058637 -0.650866 -0.929427
3    -1.229911 -0.485721  0.376231  ... -1.161414  0.274084 -0.929427
4     0.813067 -1.274939  0.376231  ... -0.885720  1.199034  0.238250
...        ...       ...       ...  ...       ...       ...       ...
1382  0.813067 -1.225613  1.786588  ...  0.492751  0.274084  0.238250
1383  0.813067 -1.176286 -1.034126  ... -0.058637 -1.575817  0.238250
1384  0.813067  1.043387  1.786588  ...  1.319834 -0.650866  0.238250
1385  0.813067 -0.140439 -1.034126  ... -0.334331  1.199034 -0.929427
1386  0.813067  0.796757  1.786588  ... -0.885720 -1.575817 -0.929427

            21        22        23        24        25        26        27
0    -0.114048 -2.171420 -2.501172  0.252482  0.040903 -1.092503  0.368270
1     0.184647  0.151871  0.336558  0.878987  0.965895  0.294582  0.982200
2    -0.282450  0.151871  0.336558 -3.122519 -1.192419 -1.092503 -1.166556
3    -0.114048  0.151871  0.336558  0.592150  0.965895  0.908019 -1.166556
4    -0.467759  0.151871  0.336558 -0.782424 -0.575758  0.655114 -0.552626
...        ...       ...       ...       ...       ...       ...       ...
1382  0.990880  0.151871  0.336558  0.053478 -0.575758 -1.092503 -0.245661
1383  0.040844  1.700732  0.336558  0.430434  0.965895  0.294582  0.982200
1384 -0.467759 -2.171420  0.336558  0.252482 -0.575758 -1.092503 -0.245661
1385  0.990880  0.151871 -1.082307  0.740892  0.657565 -1.092503  1.289165
1386 -0.467759  0.151871  1.755424 -0.174184 -0.267427  0.294582 -0.552626

[1387 rows x 28 columns]
```

In [342]:
```
xf.columns
```

Out[342]:
```
RangeIndex(start=0, stop=28, step=1)
```

In [343]:
```
column = ['age', 'daily rate', 'department',
       'distance from home', 'education', 'education field',
       'environment satisfaction', 'gender', 'hourly rate', 'job involvement',
       'job level', 'job role', 'job satisfaction', 'marital status',
       'monthly income', 'monthly rate', 'num companies worked', 'over time',
       'percent salary hike',
       'relationship satisfaction', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager']
```

In [344]:
```
xf.columns=column
```

In [345]:
```
xf.head(2)
```

Out[345]:

	age	daily rate	department	distance from home	education	education field	environment satisfaction	gender	hourly rate	job involvement
0	0.536681	0.734325	1.405373	-1.011249	-0.876177	-0.940815	-0.665328	-1.229911	1.388670	0.376231
1	1.442111	-1.307769	-0.496337	-0.145521	-1.853858	-0.940815	0.251978	0.813067	-0.239091	-1.034126

2 rows × 28 columns

In [346]:
```
xf.columns
```

Out[346]:
```
Index(['age', 'daily rate', 'department', 'distance from home', 'education',
       'education field', 'environment satisfaction', 'gender', 'hourly rate',
       'job involvement', 'job level', 'job role', 'job satisfaction',
       'marital status', 'monthly income', 'monthly rate',
       'num companies worked', 'over time', 'percent salary hike',
       'relationship satisfaction', 'stock option Level',
       'total working years', 'training times last year', 'work life balance',
       'years at company', 'years in current role',
       'years since last promotion', 'years with currmanager'],
      dtype='object')
```

In [347]:
```
xf.shape
```

Out[347]:
```
(1387, 28)
```

In [348]:
```
yf=y
```

In [349]:
```
yf.head(2)
```

Out[349]:

	attrition
0	1
1	0

In [350]:
```
yf.value_counts()
```

Out[350]:
```
attrition
0            1158
1             229
dtype: int64
```

In [ ]:
```

```

FINDING MULTICOLINEARITY

==========

In [351]:
```
# We have to find the multicollinearity between the features and to remove it we can use VIF (V
# we can not apply VIF on the TARGET COLUMN
# for apllyin VIF we have to import some libraries as follows
```

In [352]:
```
import statsmodels.api as sm
from scipy import stats
from statsmodels .stats.outliers_influence import variance_inflation_factor
```

In [353]:
```
# here we are making "def function" for calculating VIF
def calc_vif(xf):
    vif = pd.DataFrame()
    vif["FETURES"] = xf.columns
    vif["VIF FACTOR"] = [variance_inflation_factor(xf.values,i) for i in range (xf.shape[1])]
    return (vif)
```

In [354]:
```
calc_vif(xf)
# here we can find there is not much MULTICOLINEARITY in the following table.
# we already dropped irrelevant columns
# so there is no need to remove any multicolinearity
```

Out[354]:

	FETURES	VIF FACTOR
0	age	1.846164
1	daily rate	1.026975
2	department	2.089576
3	distance from home	1.017768
4	education	1.059674
5	education field	1.020008
6	environment satisfaction	1.025470
7	gender	1.024081
8	hourly rate	1.023153
9	job involvement	1.015382
10	job level	7.852482
11	job role	1.991356
12	job satisfaction	1.022241
13	marital status	1.848049
14	monthly income	7.651213
15	monthly rate	1.017877
16	num companies worked	1.291587
17	over time	1.026692
18	percent salary hike	1.013236
19	relationship satisfaction	1.021178
20	stock option Level	1.819702
21	total working years	3.820242
22	training times last year	1.026929
23	work life balance	1.018520
24	years at company	4.072469
25	years in current role	2.946505
26	years since last promotion	1.365335
27	years with currmanager	2.987948

In [ ]:

RESAMPLING TECHNIQUE (APPLYING SMOTE)
==========

In [355]:
```
# Here we know that our Target Column is a Categorical column. which is having values from 0-1
# so we have to chek the distribution of values are equal or not, offcourse i would be not, so
#  'equally balanced distributed' for better results.

# SOLVING CLASS IMMBALANCE PROBLEM BY SMOTE TECHNIQUE.
```

In [356]:
```
yf.value_counts()
# here we can clearly see the imbalance in TARGET COULUMN
# for better result and performance of model, we have to first make it balanced.
```

Out[356]:
```
attrition
0            1158
1             229
dtype: int64
```

In [357]:
```
# To solve this prolem we need import SMOTE LIBRARY from the IMBLEARN.
```

In [358]:
```
from imblearn.over_sampling import SMOTE
```

In [359]:
```
smt = SMOTE()
```

In [360]:
```
trainx, trainy = smt.fit_resample(xf,yf)
```

In [361]:
```
trainy.value_counts()
# here as you can see below the immbalancenes is cleared now.
# and now our Target Column Categories are BALANCED NOW.
```

Out[361]:
```
attrition
0            1158
1            1158
dtype: int64
```

In [362]:
```
trainx.shape
```

Out[362]:
```
(2316, 28)
```

In [363]:
```
trainy.shape
```

Out[363]:
```
(2316, 1)
```

========== UPTO HERE EDA AND OTHER TECHINIQUES ARE COMPLETED
==========

========== NOW WE NEED TO APPLY ML MODELS
==========

In [364]:
```
trainy.nunique()
```

Out[364]:
```
attrition    2
dtype: int64
```

In [365]:
```
# here above as we know that our target column is CATEGORICAL and having 2 values = 0, 1
# therefore it as an CLASSIFICATION PROBLEM. and we need to apply classification Ml Algorithm
```

In [366]:
```
#  Applying TRAIN_TEST_SPLIT ====>>>
#  IMPORTING SOME IMPORTANT REQUIIRED LIBRARIES
```

In [367]:
```
from sklearn.model_selection import train_test_split
```

In [368]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [369]:
```
import sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
```

In [370]:
```
lg = LogisticRegression()
gnb = GaussianNB()
svc = SVC()
dtc = DecisionTreeClassifier()
knn = KNeighborsClassifier()
```

In [371]:
```
model = [lg,gnb,svc,dtc,knn]
```

In [372]:
```
#  1) Applying LOGISTIC REGRESSION MODEL & FINDING BEST RANDOM STATE ===>
```

In [375]:
```
lg.fit(x_train,y_train)
lg.score(x_train,y_train)
lg_pred = lg.predict(x_train)

print('Accuracy Score of ', lg,'is:')
print (accuracy_score(y_train,lg_pred))

print(confusion_matrix(y_train,lg_pred))
print(classification_report(y_train,lg_pred))
print('\n')

# here with Logistic Regression Model , we are getting accuracy of = 80%
```

```
Accuracy Score of  LogisticRegression() is:
0.7985961123110151
[[734 189]
 [184 745]]
              precision    recall  f1-score   support

           0       0.80      0.80      0.80       923
           1       0.80      0.80      0.80       929

    accuracy                           0.80      1852
   macro avg       0.80      0.80      0.80      1852
weighted avg       0.80      0.80      0.80      1852
```

In [376]:
```
maxaccu = 0
maxrs = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(trainx,trainy,test_size=0.20,random_state=
    lg = LogisticRegression()
    lg.fit(x_train,y_train)
    pred = lg.predict(x_test)
    acc = accuracy_score(y_test,pred)

    if acc > maxaccu :
        maxaccu = acc
        maxrs = i

print ("Best accuracy is",maxaccu, "at random state", maxrs)

# here the with LOGISTIC REGRESSION MODEL- Accuracy is =84%  &  Best Random State is = 194
# here we can see that the ACCURACY is also increased with finding BEST RANDOM STATE for the mo
```

```
Best accuracy is 0.8362068965517241 at random state 194
```

In [ ]:
```

```

In [377]:
```
# 2) Applying GGAUSSIAN-NB Model & FINDING BEST RANDOM STATE FOR THE MODEL======>>>
```

In [378]:
```
gnb.fit(x_train,y_train)
gnb.score(x_train,y_train)
gnb_pred = gnb.predict(x_train)

print('Accuracy Score of ', gnb,'is:')
print (accuracy_score(y_train,gnb_pred))

print(confusion_matrix(y_train,gnb_pred))
print(classification_report(y_train,gnb_pred))
print('\n')

# here with Gaussian-NB Model , we are getting accuracy of = 72 %
```

```
Accuracy Score of  GaussianNB() is:
0.7240820734341252
[[630 293]
 [218 711]]
              precision    recall  f1-score   support

           0       0.74      0.68      0.71       923
           1       0.71      0.77      0.74       929

    accuracy                           0.72      1852
   macro avg       0.73      0.72      0.72      1852
weighted avg       0.73      0.72      0.72      1852
```

In [379]:
```
maxaccu = 0
maxrs = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(trainx,trainy,test_size=0.20,random_state=
    gnb = GaussianNB()
    gnb.fit(x_train,y_train)
    pred = gnb.predict(x_test)
    acc = accuracy_score(y_test,pred)

    if acc > maxaccu :
        maxaccu = acc
        maxrs = i

print ("Best accuracy is",maxaccu, "at random state", maxrs)

# here the with GAussian NB-MODEL- Accuracy is =77 %  &  Best Random State is = 172
# here we find that ACCURACY is also increased with FINDING BEST RANDOM STATE
```

```
Best accuracy is 0.7672413793103449 at random state 172
```

In [ ]:
```

```

In [380]:
```
# 3) Applying Support Vector Classifier Model & Finding Best Random State For it =====>>>>
```

In [381]:
```
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
svc_pred = svc.predict(x_train)

print('Accuracy Score of ', svc,'is:')
print (accuracy_score(y_train,svc_pred))

print(confusion_matrix(y_train,svc_pred))
print(classification_report(y_train,svc_pred))
print('\n')

# here with Support Vector Classifier Model , we are getting accuracy of = 96 %
```

```
Accuracy Score of  SVC() is:
0.9568034557235421
[[871  52]
 [ 28 901]]
              precision    recall  f1-score   support

           0       0.97      0.94      0.96       923
           1       0.95      0.97      0.96       929

    accuracy                           0.96      1852
   macro avg       0.96      0.96      0.96      1852
weighted avg       0.96      0.96      0.96      1852
```

In [382]:
```
maxaccu = 0
maxrs = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(trainx,trainy,test_size=0.20,random_state=
    svc = SVC()
    svc.fit(x_train,y_train)
    pred = svc.predict(x_test)
    acc = accuracy_score(y_test,pred)

    if acc > maxaccu :
        maxaccu = acc
        maxrs = i

print ("Best accuracy is",maxaccu, "at random state", maxrs)

# here the with Support Vector Classifier-MODEL- Accuracy is =95 %  &  Best Random State is = 7
# here we find that ACCURACY is changed with FINDING BEST RANDOM STATE
```

```
Best accuracy is 0.9482758620689655 at random state 75
```

In [ ]:
```

```

In [383]:
```
# 4) Applying Decision Tree Classifier Model & Findig Best Random State for it =========>>>>
```

In [384]:
```
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
dtc_pred = dtc.predict(x_train)

print('Accuracy Score of ', dtc,'is:')
print (accuracy_score(y_train,dtc_pred))

print(confusion_matrix(y_train,dtc_pred))
print(classification_report(y_train,dtc_pred))
print('\n')

# here with Decision Tree Classifier Model , we are getting accuracy of = 100 %
```

```
Accuracy Score of  DecisionTreeClassifier() is:
1.0
[[923   0]
 [  0 929]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       923
           1       1.00      1.00      1.00       929

    accuracy                           1.00      1852
   macro avg       1.00      1.00      1.00      1852
weighted avg       1.00      1.00      1.00      1852
```

In [385]:
```
maxaccu = 0
maxrs = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(trainx,trainy,test_size=0.20,random_state=
    dtc = DecisionTreeClassifier()
    dtc.fit(x_train,y_train)
    pred = dtc.predict(x_test)
    acc = accuracy_score(y_test,pred)

    if acc > maxaccu :
        maxaccu = acc
        maxrs = i

print ("Best accuracy is",maxaccu, "at random state", maxrs)

# here the with Decision Tree Classifier-MODEL- Accuracy is =89 %  &  Best Random State is = 44
# here we find that ACCURACY is changed with FINDING BEST RANDOM STATE
```

```
Best accuracy is 0.8943965517241379 at random state 44
```

In [ ]:
```
```

In [386]:
```
# 5) Applying KNearest Neighbour Model & Finding Best Random State For it =====>>>>
```

In [387]:
```
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
knn_pred = knn.predict(x_train)

print('Accuracy Score of ', knn,'is:')
print (accuracy_score(y_train,knn_pred))

print(confusion_matrix(y_train,knn_pred))
print(classification_report(y_train,knn_pred))
print('\n')

# here with KNeighbor Classifier Model , we are getting accuracy of = 86 %
```

```
Accuracy Score of  KNeighborsClassifier() is:
0.8612311015118791
[[673 250]
 [  7 922]]
              precision    recall  f1-score   support

           0       0.99      0.73      0.84       923
           1       0.79      0.99      0.88       929

    accuracy                           0.86      1852
   macro avg       0.89      0.86      0.86      1852
weighted avg       0.89      0.86      0.86      1852
```

In [388]:
```
maxaccu = 0
maxrs = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(trainx,trainy,test_size=0.20,random_state=
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    pred = knn.predict(x_test)
    acc = accuracy_score(y_test,pred)

    if acc > maxaccu :
        maxaccu = acc
        maxrs = i

print ("Best accuracy is",maxaccu, "at random state", maxrs)

# here the with KNeighbors Classifier-MODEL- Accuracy is =86 %  &  Best Random State is = 106
# here we find that ACCURACY is changed with FINDING BEST RANDOM STATE
```

```
Best accuracy is 0.853448275862069 at random state 106
```

In [ ]:

In [389]:
```
#  After Applying Above 5 Classification ML Models we are getting following accuracies :
# lg = LogisticRegression()        = accuracy-84 %   with  Best Random State- 194
# gnb = GaussianNB()               = accuracy-77 %   with  Best Random State- 172
# svc = SVC()                      = accuracy-95%   with  Best Random State- 75
# dtc = DecisionTreeClassifier()   = accuracy-90 %   with  Best Random State- 44
# knn = KNeighborsClassifier()     = accuracy-86 %   with  Best Random State- 106

#  so from the above accuracies we can find that the best working model is SVC() with = 94 % ac
# so we can apply SVC() as a FINAL MODEL
```

In [ ]:

========APPLYING SVC (SUPPRT VECTOR CLASSIFIER AS A FINAL MODEL) ==============

In [390]:
```
final_model = SVC()
```

In [392]:
```
x_train,x_test,y_train,y_test = train_test_split(trainx,trainy,test_size=0.20,random_state=75)
final_model.fit(x_train,y_train)
final_model.score(x_train,y_train)
final_model_pred = final_model.predict(x_test)
print(accuracy_score(y_test,final_model_pred))
print(confusion_matrix(y_test,final_model_pred))
print(classification_report(y_test,final_model_pred))

# here KNEIGHBORS CLASSIFIER as FINAL MODEL with ACCURACY OF = 94 %
```

```
0.9482758620689655
[[230  13]
 [ 11 210]]
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       243
           1       0.94      0.95      0.95       221

    accuracy                           0.95       464
   macro avg       0.95      0.95      0.95       464
weighted avg       0.95      0.95      0.95       464
```

In [ ]:

In [393]:
```
# Making 'def' function to CHECK / VERIFY samples :
```

In [394]:
```
xf.shape
```

Out[394]: (1387, 28)

In [395]:
```
def pred_func(at):
    at= at.reshape(1,28)
    attrition = final_model.predict(at)
    print(attrition)

    if attrition == 0:
        print("No Attrition")
    elif (attrition == 1):
        print ("Yes Attrition")
    else:
        print('Not Found')
# making 'def' function to predict Attrition .
```

In [ ]:

In [396]:
```
pd.set_option('display.max_columns', None)
xf

# here by making 'display.amx_columns' we can see all the 28 columns of dataset.
```

Out[396]:

	age	daily rate	department	distance from home	education	education field	environment satisfaction	gender	hourly rate	involvem
0	0.536681	0.734325	1.405373	-1.011249	-0.876177	-0.940815	-0.665328	-1.229911	1.388670	0.3762
1	1.442111	-1.307769	-0.496337	-0.145521	-1.853858	-0.940815	0.251978	0.813067	-0.239091	-1.034
2	0.083966	1.406752	-0.496337	-0.887573	-0.876177	1.305159	1.169285	0.813067	1.290017	-1.034
3	-0.368749	1.453896	-0.496337	-0.763898	1.079185	-0.940815	1.169285	-1.229911	-0.485721	0.3762
4	-1.047821	-0.533609	-0.496337	-0.887573	-1.853858	0.556501	-1.582635	0.813067	-1.274939	0.3762
...	...	...	...	...	...	...	...	...	...	
1382	-0.029213	0.193406	-0.496337	1.709609	-0.876177	0.556501	0.251978	0.813067	-1.225613	1.7865
1383	0.310324	-0.479021	-0.496337	-0.392872	-1.853858	0.556501	1.169285	0.813067	-1.176286	-1.034
1384	-1.047821	-1.615447	-0.496337	-0.640223	0.101504	-0.940815	-0.665328	0.813067	1.043387	1.7865
1385	1.442111	0.538304	1.405373	-0.887573	0.101504	0.556501	1.169285	0.813067	-0.140439	-1.034
1386	-0.255570	-0.441802	-0.496337	-0.145521	0.101504	0.556501	-0.665328	0.813067	0.796757	1.7865

1387 rows × 28 columns

In [397]:
```
yf
```

Out[397]:

	attrition
0	1
1	0
2	1
3	0
4	0
...	...
1465	0
1466	0
1467	0
1468	0
1469	0

1387 rows × 1 columns

In [ ]:
```

```

In [398]:
```
# Test Sample 1 (taking data from row no. '0')
```

In [399]:
```
at= np.array([0.536681,0.734325,1.405373,-1.011249,-0.876177,-0.940815,-0.665328,-1.229911,1.3
pred_func(at)

# here below we can see that our model is predictiong [1] Yes Attrition, which as 100% matched
```

```
[0]
No Attrition
```

In [ ]:
```

```

In [400]:
```
# Test Sample 2 (taking data from row no '3')
```

In [401]:
```
at= np.array([-0.368749,1.453896,-0.496337,-0.763898,1.079185,-0.940815,1.169285,-1.229911,-0.4
pred_func(at)

# here below we can see that our model is predictiong "[0] No Attrition ", which as 100% matche
```

```
[0]
No Attrition
```

In [ ]:
```

```

SAVING MODEL

==========

In [402]:
```
import pickle
```

In [403]:
```
file_name = 'Hr_Analytics.pkl'
pickle.dump(final_model,open(file_name,'wb'))
```

In [ ]:

========== FINISHED
==========

In [ ]: