Recruiting and retaining drivers is seen by industry watchers as a tough battle for Ola. Churn among drivers is high and it's very easy for drivers to stop working for the service on the fly or jump to Uber depending on the rates.

As the companies get bigger, the high churn could become a bigger problem. To find new drivers, Ola is casting a wide net, including people who don't have cars for jobs. But this acquisition is really costly. Losing drivers frequently impacts the morale of the organization and acquiring new drivers is more expensive than retaining existing ones.

Problem Statement :

You are working as a data scientist with the Analytics Department of Ola, focused on driver team attrition. You are provided with the monthly information for a segment of drivers for 2019 and 2020 and tasked to predict whether a driver will be leaving the company or not based on their attributes like :

- Demographics (city, age, gender etc.)
- Tenure information (joining date, Last Date)
- Historical data regarding the performance of the driver (Quarterly rating, Monthly business acquired, grade, Income)

Data Description :

- MMMM-YY : Reporting Date (Monthly)
- Driver_ID : Unique id for drivers
- Age : Age of the driver
- Gender : Gender of the driver – Male : 0, Female: 1
- City : City Code of the driver
- Education_Level : Education level – 0 for 10+ ,1 for 12+ ,2 for graduate
- Income : Monthly average Income of the driver
- Date Of Joining : Joining date for the driver
- LastWorkingDate : Last date of working for the driver
- Joining Designation : Designation of the driver at the time of joining
- Grade : Grade of the driver at the time of reporting
- Total Business Value : The total business value acquired by the driver in a month (negative business indicates cancellation/refund or car EMI adjustments)
- Quarterly Rating : Quarterly rating of the driver: 1,2,3,4,5 (higher is better)

Import the Libraries :

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns' , None)
pd.set_option('display.max_rows' , None)
```

In [2]:

```
import matplotlib as mpl
import plotly.graph_objs as go
import matplotlib.patches as mpatches
import plotly.express as px
from plotly import tools
from plotly.subplots import make_subplots
from plotly.offline import iplot
```

In [3]:

```
df = pd.read_csv("ola_driver_scaler.csv")
df.head()
```

Out[3]:

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quarterly Rating
0	0	01/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	2381060	2
1	1	02/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	-665480	2
2	2	03/01/19	1	28.0	0.0	C23	2	57387	24/12/18	03/11/19	1	1	0	2
3	3	11/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1
4	4	12/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1

In [4]:

```
df.ndim
```

Out[4]:

```
2
```

In [5]:

```
df.shape
```

Out[5]:

```
(19104, 14)
```

In [6]:

```
df.size
```

Out[6]:

```
267456
```

In [7]:

```
df = df.drop(['Unnamed: 0'], axis=1)
df.head()
```

Out[7]:

	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quarterly Rating
0	01/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	2381060	2
1	02/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	-665480	2
2	03/01/19	1	28.0	0.0	C23	2	57387	24/12/18	03/11/19	1	1	0	2
3	11/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1
4	12/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1

In [8]:

```
df.info()
```

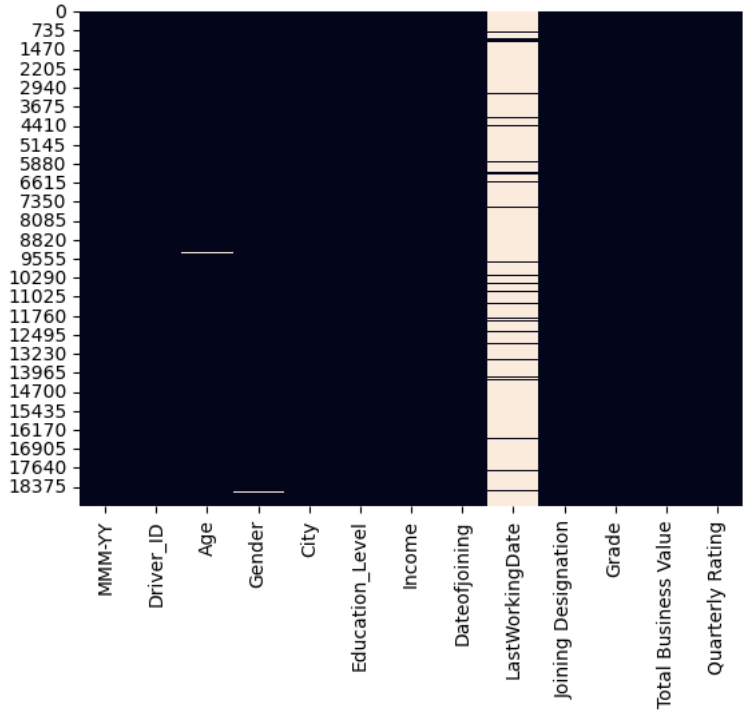
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   MMM-YY                19104 non-null  object 
 1   Driver_ID             19104 non-null  int64  
 2   Age                   19043 non-null  float64
 3   Gender                19052 non-null  float64
 4   City                  19104 non-null  object 
 5   Education_Level       19104 non-null  int64  
 6   Income                19104 non-null  int64  
 7   Dateofjoining         19104 non-null  object 
 8   LastWorkingDate       1616 non-null   object 
 9   Joining Designation   19104 non-null  int64  
 10  Grade                 19104 non-null  int64  
 11  Total Business Value  19104 non-null  int64  
 12  Quarterly Rating      19104 non-null  int64  
dtypes: float64(2), int64(7), object(4)
memory usage: 1.9+ MB
```

In [9]:

```
sns.heatmap(df.isnull() ,cbar=False)
```

Out[9]:

<AxesSubplot:>

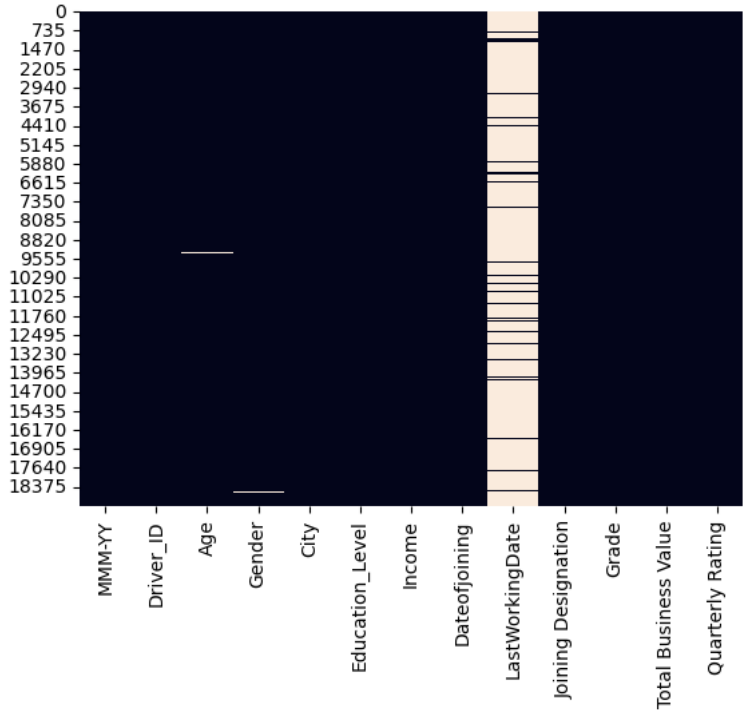


In [10]:

```
sns.heatmap(df.isna() ,cbar=False)
```

Out[10]:

<AxesSubplot:>



In [11]:

```
df.isnull().sum()/len(df.index)*100
```

Out[11]:

```
MMM-YY                  0.000000
Driver_ID               0.000000
Age                     0.319305
Gender                  0.272194
City                    0.000000
Education_Level         0.000000
Income                  0.000000
Dateofjoining           0.000000
LastWorkingDate        91.541039
Joining Designation     0.000000
Grade                   0.000000
Total Business Value    0.000000
Quarterly Rating        0.000000
dtype: float64
```

In [12]:

```
df.columns
```

Out[12]:

```
Index(['MMM-YY', 'Driver_ID', 'Age', 'Gender', 'City', 'Education_Level',
       'Income', 'Dateofjoining', 'LastWorkingDate', 'Joining Designation',
       'Grade', 'Total Business Value', 'Quarterly Rating'],
      dtype='object')
```

In [13]:

```
df.index
```

Out[13]:

```
RangeIndex(start=0, stop=19104, step=1)
```

In [14]:

```
df.dtypes
```

Out[14]:

```
MMM-YY                  object
Driver_ID                int64
Age                    float64
Gender                 float64
City                    object
Education_Level          int64
Income                   int64
Dateofjoining           object
LastWorkingDate         object
Joining Designation      int64
Grade                    int64
Total Business Value     int64
Quarterly Rating         int64
dtype: object
```

In [15]:

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

Out[15]:

```
array([28., 31., 43., 29., 34., 35., 30., 39., 42., 27., 26., nan, 33.,
       40., 41., 32., 22., 44., 36., 21., 49., 37., 38., 46., 47., 48.,
       25., 24., 45., 51., 52., 23., 50., 53., 54., 55., 58.])
```

In [16]:

```
categorical_columns = ['Age' , 'Gender' , 'Education_Level' , 'Income' , 'Joining Designation' , 'Grade']
```

In [17]:

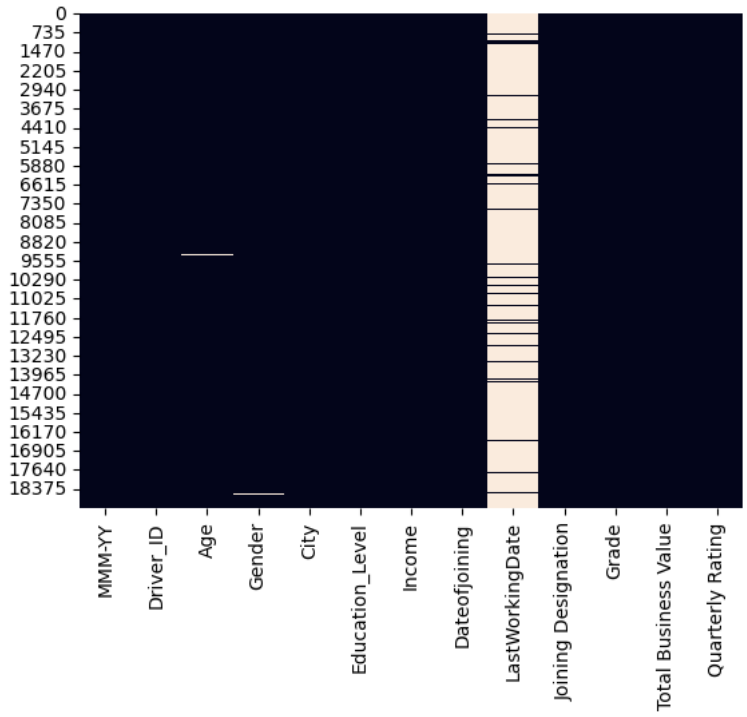
```
# df['Attrition'] = df['LastWorkingDate'].fillna(0)
# df['Attrition'] = df['LastWorkingDate'].notnull()
# df['Attrition'] = df['Attrition'].map({False:0, True: 1})
```

In [18]:

```
sns.heatmap(df.isnull() ,cbar=False)
```

Out[18]:

<AxesSubplot:>



In [19]:

```
df.head()
```

Out[19]:

	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quarterly Rating
0	01/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	2381060	2
1	02/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	-665480	2
2	03/01/19	1	28.0	0.0	C23	2	57387	24/12/18	03/11/19	1	1	0	2
3	11/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1
4	12/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1

In [20]:

```
df['MMM-YY'] = pd.to_datetime(df['MMM-YY'])
df['Dateofjoining'] = pd.to_datetime(df['Dateofjoining'])
df['LastWorkingDate'] = pd.to_datetime(df['LastWorkingDate'])
```

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   MMM-YY               19104 non-null  datetime64[ns]
 1   Driver_ID            19104 non-null  int64         
 2   Age                  19043 non-null  float64       
 3   Gender               19052 non-null  float64       
 4   City                 19104 non-null  object        
 5   Education_Level      19104 non-null  int64         
 6   Income               19104 non-null  int64         
 7   Dateofjoining        19104 non-null  datetime64[ns]
 8   LastWorkingDate      1616 non-null   datetime64[ns]
 9   Joining Designation  19104 non-null  int64         
 10  Grade                19104 non-null  int64         
 11  Total Business Value 19104 non-null  int64         
 12  Quarterly Rating     19104 non-null  int64         
dtypes: datetime64[ns](3), float64(2), int64(7), object(1)
memory usage: 1.9+ MB
```

In [22]:

```
categorical_features = list(df.select_dtypes('object').columns)
```

In [23]:

```
for i in categorical_features:
    print('Unique Values in {0} are {1}'.format(i,df[i].unique()))
```

```
Unique Values in City are ['C23' 'C7' 'C13' 'C9' 'C11' 'C2' 'C19' 'C26' 'C20' 'C17' 'C29' 'C10'
 'C24' 'C14' 'C6' 'C28' 'C5' 'C18' 'C27' 'C15' 'C8' 'C25' 'C21' 'C1' 'C4'
 'C3' 'C16' 'C22' 'C12']
```

In [24]:

```
df["Driver_ID"].value_counts().head(20)
```

Out[24]:

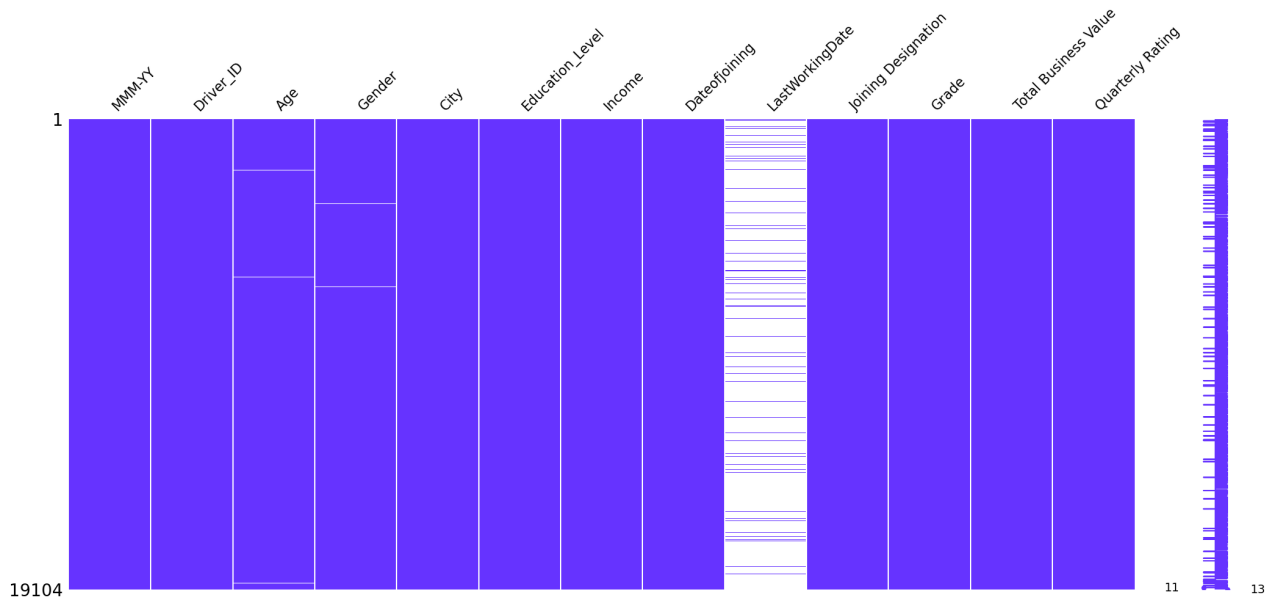
```
2110    24
2617    24
1623    24
1642    24
1644    24
1655    24
1657    24
1662    24
1664    24
2259    24
2615    24
486     24
1510    24
1670    24
2257    24
1678    24
1679    24
2625    24
2255    24
450     24
Name: Driver_ID, dtype: int64
```

In [25]:

```
import missingno as mn
mn.matrix(df,color=(0.40,0.20,1))
```

Out[25]:

```
<AxesSubplot:>
```



In [26]:

```
#PERCENTAGE OF THE MISSING VALUES - DATAFRAME.....
def missing_data(df):
    total = df.isnull().sum().sort_values(ascending = False)
    Percentage = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending = False)
    return pd.concat([total, Percentage], axis=1, keys=['Total', 'Percentage'])
```

In [27]:

```
missing_data(df).style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-color": "white","font-size":"11.5pt",'wi
```

Out[27]:

	Total	Percentage
LastWorkingDate	17488	91.541039
Age	61	0.319305
Gender	52	0.272194
MMM-YY	0	0.000000
Driver_ID	0	0.000000
City	0	0.000000
Education_Level	0	0.000000
Income	0	0.000000
Dateofjoining	0	0.000000
Joining Designation	0	0.000000
Grade	0	0.000000
Total Business Value	0	0.000000
Quarterly Rating	0	0.000000

In [28]:

```
df[df.duplicated()]
```

Out[28]:

MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quarterly Rating

In [29]:

```
round(df.describe(exclude = ['object',"datetime64[ns]"]), 2)
```

Out[29]:

	Driver_ID	Age	Gender	Education_Level	Income	Joining Designation	Grade	Total Business Value	Quarterly Rating
count	19104.00	19043.00	19052.00	19104.00	19104.00	19104.00	19104.00	19104.00	19104.00
mean	1415.59	34.67	0.42	1.02	65652.03	1.69	2.25	571662.07	2.01
std	810.71	6.26	0.49	0.80	30914.52	0.84	1.03	1128312.22	1.01
min	1.00	21.00	0.00	0.00	10747.00	1.00	1.00	-6000000.00	1.00
25%	710.00	30.00	0.00	0.00	42383.00	1.00	1.00	0.00	1.00
50%	1417.00	34.00	0.00	1.00	60087.00	1.00	2.00	250000.00	2.00
75%	2137.00	39.00	1.00	2.00	83969.00	2.00	3.00	699700.00	3.00
max	2788.00	58.00	1.00	2.00	188418.00	5.00	5.00	33747720.00	4.00

In [30]:

```
round(df.describe(exclude = ["datetime64[ns]",'float', 'int64']),2)
```

Out[30]:

	City
count	19104
unique	29
top	C20
freq	1008

In [31]:

```
round(df.describe(exclude = ["object",'float', 'int64']),2)
```

Out[31]:

	MMM-YY	Dateofjoining	LastWorkingDate
count	19104	19104	1616
unique	24	869	493
top	2019-01-01 00:00:00	2015-07-23 00:00:00	2020-07-29 00:00:00
freq	1022	192	70
first	2019-01-01 00:00:00	2013-04-01 00:00:00	2018-12-31 00:00:00
last	2020-12-01 00:00:00	2020-12-28 00:00:00	2020-12-28 00:00:00

In [32]:

```
df.head()
```

Out[32]:

	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quarterly Rating
0	2019-01-01	1	28.0	0.0	C23	2	57387	2018-12-24	NaT	1	1	2381060	2
1	2019-02-01	1	28.0	0.0	C23	2	57387	2018-12-24	NaT	1	1	-665480	2
2	2019-03-01	1	28.0	0.0	C23	2	57387	2018-12-24	2019-03-11	1	1	0	2
3	2020-11-01	2	31.0	0.0	C7	2	67016	2020-11-06	NaT	2	2	0	1
4	2020-12-01	2	31.0	0.0	C7	2	67016	2020-11-06	NaT	2	2	0	1

In [33]:

```
df_1 = pd.DataFrame()
```

In [34]:

```
df_1['Driver_ID'] = df['Driver_ID'].unique()

#Setting age of the employee which takes the maximum age of the employee
df_1['Age'] = list(df.groupby('Driver_ID',axis=0).max('MMM-YY')['Age'])

#Setting gender of the employee
df_1['Gender'] = list(df.groupby('Driver_ID').agg({'Gender':'first'})['Gender'])

#Setting the city of the employee
df_1['City'] = list(df.groupby('Driver_ID').agg({'City':'first'})['City'])

#Setting the education of the employee
df_1['Education'] = list(df.groupby('Driver_ID').agg({'Education_Level':'last'})['Education_Level'])

#Setting the salary of the employee for one month
df_1['Income'] = list(df.groupby('Driver_ID').agg({'Income':'last'})['Income'])

#Setting the joining designtion of the employee
df_1['Joining_Designation'] = list(df.groupby('Driver_ID').agg({'Joining Designation':'last'})['Joining Designation'])

#Setting the designtion of the employee at the time of reporting
df_1['Designation'] = list(df.groupby('Driver_ID').agg({'Grade':'last'})['Grade'])
```

In [35]:

```
z = df.groupby('Driver_ID',axis=0).sum('Total Business Value')['Total Business Value']
z.head(5)
```

Out[35]:

```
Driver_ID
1    1715580
2          0
4     350000
5     120360
6    1265000
Name: Total Business Value, dtype: int64
```

In [36]:

```
y = df.groupby('Driver_ID',axis=0).sum({'Quarterly Rating':'last'})['Quarterly Rating']
y.head(5)
```

Out[36]:

```
Driver_ID
1    6
2    2
4    5
5    3
6    8
Name: Quarterly Rating, dtype: int64
```

In [37]:

```
df_1['Total_Business_Value'] = list(z)
```

In [38]:

```
df_1['Last_Quarterly_Rating'] = list(y)
```

In [39]:

```
#Creating a column which tells if the quarterly rating has increased for that employee
#for those whose quarterly rating has increased we assign the value 1

#Quarterly rating at the beginning
qrf = df.groupby('Driver_ID').agg({'Quarterly Rating':'first'})

#Quarterly rating at the end
qrl = df.groupby('Driver_ID').agg({'Quarterly Rating':'last'})
#The dataset which has the employee ids and a bollean value which tells if the rating has increased
qr = (qrl['Quarterly Rating'] > qrf['Quarterly Rating']).reset_index()

#the employee ids whose rating has increased
driverid = qr[qr['Quarterly Rating']==True]['Driver_ID']

qri = []
for i in df_1['Driver_ID']:
    if i in driverid:
        qri.append(1)
    else:
        qri.append(0)

df_1['Quarterly_Rating_Increased'] = qri
```

In [40]:

```
#Creating a column called target which tells if the person has left the company
#persons who have a last working date will have the value 1

#The dataset which has the employee ids and specifies if last working date is null
lwr = (df.groupby('Driver_ID').agg({'LastWorkingDate':'last'})['LastWorkingDate'].isna()).reset_index()

#The employee ids who do not have last working date
driverid = list(lwr[lwr['LastWorkingDate']==True]['Driver_ID'])

Attrition = []
for i in df_1['Driver_ID']:
    if i in driverid:
        Attrition.append(0)
    elif i not in driverid:
        Attrition.append(1)

df_1['Attrition'] = Attrition
```

In [41]:

```
df_1.head(5)
```

Out[41]:

	Driver_ID	Age	Gender	City	Education	Income	Joining_Designation	Designation	Total_Business_Value	Last_Quarterly_Rating	Quarterly_Rating_Incr
0	1	28.0	0.0	C23	2	57387	1	1	1715580	6	
1	2	31.0	0.0	C7	2	67016	2	2	0	2	
2	4	43.0	0.0	C13	2	65603	2	2	350000	5	
3	5	29.0	0.0	C9	0	46368	1	1	120360	3	
4	6	31.0	1.0	C11	1	78728	3	3	1265000	8	

In [42]:

```
#Created a new column called "promotion" because there is a chances of employee leaving company due to not getting promoted...may be!
df_1['Promotion'] = np.where(df_1['Designation'] > df_1['Joining_Designation'] , 1 , 0)
```

In [43]:

```
df_1.head()
```

Out[43]:

	Driver_ID	Age	Gender	City	Education	Income	Joining_Designation	Designation	Total_Business_Value	Last_Quarterly_Rating	Quarterly_Rating_Incr
0	1	28.0	0.0	C23	2	57387	1	1	1715580	6	
1	2	31.0	0.0	C7	2	67016	2	2	0	2	
2	4	43.0	0.0	C13	2	65603	2	2	350000	5	
3	5	29.0	0.0	C9	0	46368	1	1	120360	3	
4	6	31.0	1.0	C11	1	78728	3	3	1265000	8	

In [44]:

```
df['LastWorkingDate'].fillna(0,inplace = True)
df['LastWorkingDate'] = df['LastWorkingDate'].apply(lambda x:0 if x==0 else 1)
```

In [45]:

```
old_df = df[['Driver_ID','MMM-YY','City','Dateofjoining']]
```

In [46]:

```
new_df = df[['Education_Level','Age','Gender','Income','Joining Designation','Grade','Total Business Value','Quarterly Rating','LastWorkir
```

In [47]:

```
from sklearn.impute import KNNImputer
from sklearn.preprocessing import MinMaxScaler
im = KNNImputer(n_neighbors=5)
new_df = pd.DataFrame(im.fit_transform(new_df),columns=new_df.columns)
```

In [48]:

```
frames= [old_df,new_df]
df = pd.concat(frames,axis=1)
```

Univariate Analysis on Numerical Features :

In [49]:

```
# list of numerical variables.....
numerical_features = [feature for feature in df_1.columns if ((df_1[feature].dtypes != 'O') & (df_1[feature].dtypes != 'datetime64[ns]'))]

print('Number of numerical variables: ', len(numerical_features))
print('\n')
print('Numberical Variables Column: ',numerical_features)
print('\n')
# visualise the numerical variables.....
df_1[numerical_features].head().style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-color": "white","font-size
```

```
Number of numerical variables:  12


Numberical Variables Column:  ['Driver_ID', 'Age', 'Gender', 'Education', 'Income', 'Joining_Designation', 'Designation',
'Total_Business_Value', 'Last_Quarterly_Rating', 'Quarterly_Rating_Increased', 'Attrition', 'Promotion']
```

Out[49]:

	Driver_ID	Age	Gender	Education	Income	Joining_Designation	Designation	Total_Business_Value	Last_Quarterly_Rating	Quarterly_Rati
0	1	28.000000	0.000000	2	57387	1	1	1715580	6	
1	2	31.000000	0.000000	2	67016	2	2	0	2	
2	4	43.000000	0.000000	2	65603	2	2	350000	5	
3	5	29.000000	0.000000	0	46368	1	1	120360	3	
4	6	31.000000	1.000000	1	78728	3	3	1265000	8	

In [50]:

```
numerical_features
```

Out[50]:

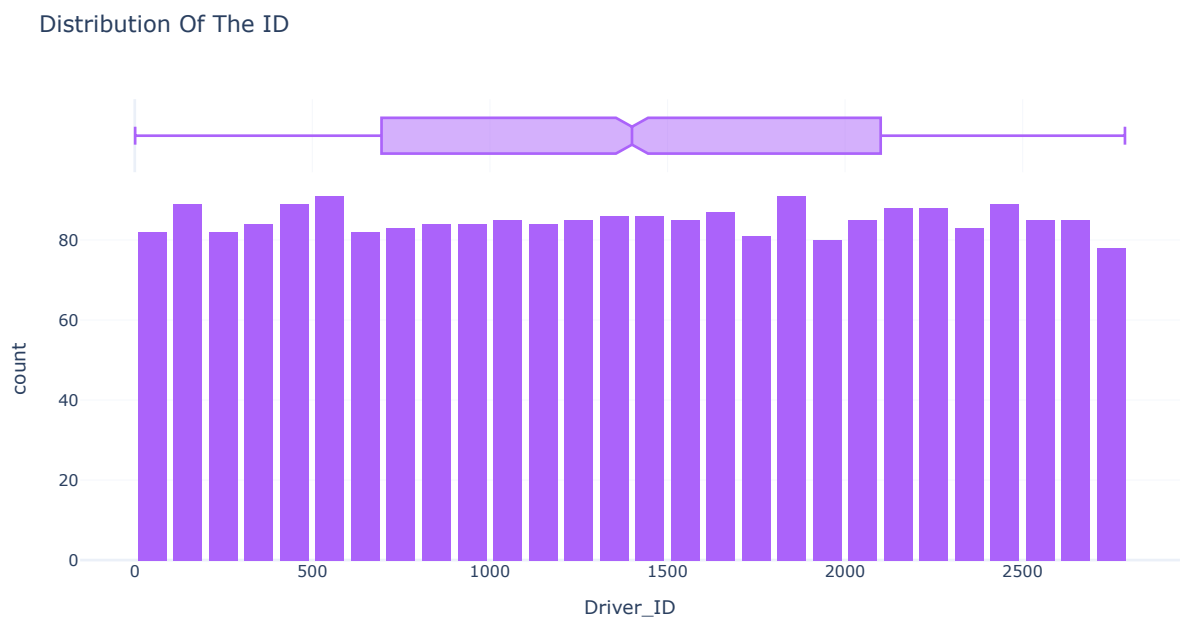
```
['Driver_ID',
 'Age',
 'Gender',
 'Education',
 'Income',
 'Joining_Designation',
 'Designation',
 'Total_Business_Value',
 'Last_Quarterly_Rating',
 'Quarterly_Rating_Increased',
 'Attrition',
 'Promotion']
```

In [51]:

```
## Lets analyse the numerical values by creating histograms to understand the distribution
```

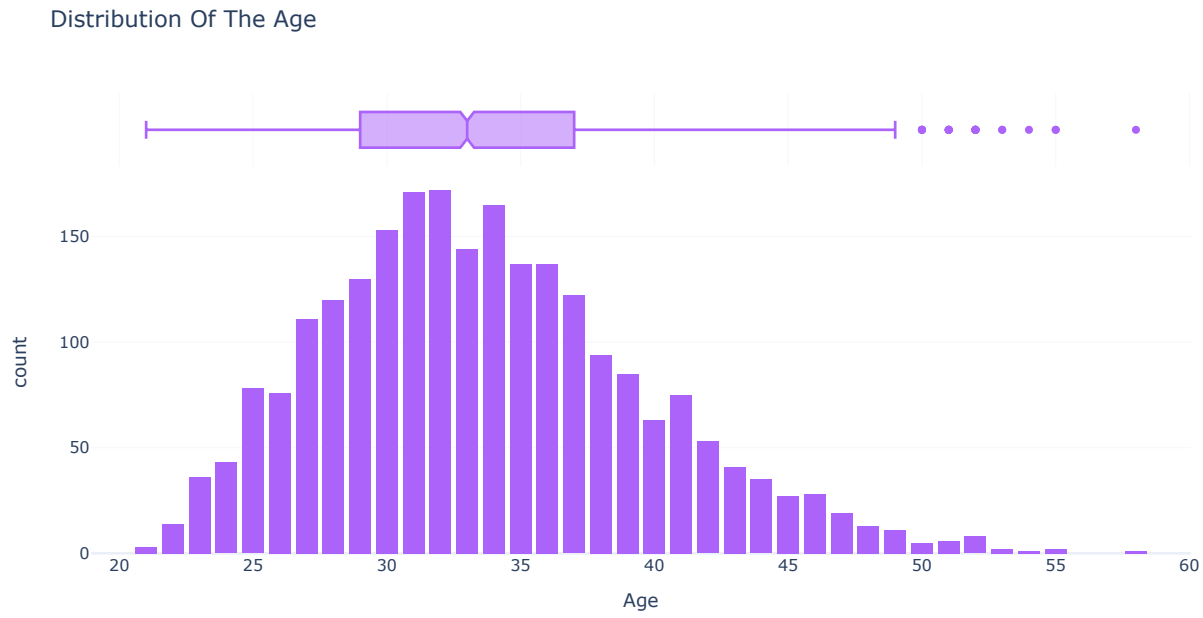
In [52]:

```
data = df_1.copy()
fig = px.histogram(df_1, x = 'Driver_ID' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The ID")
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



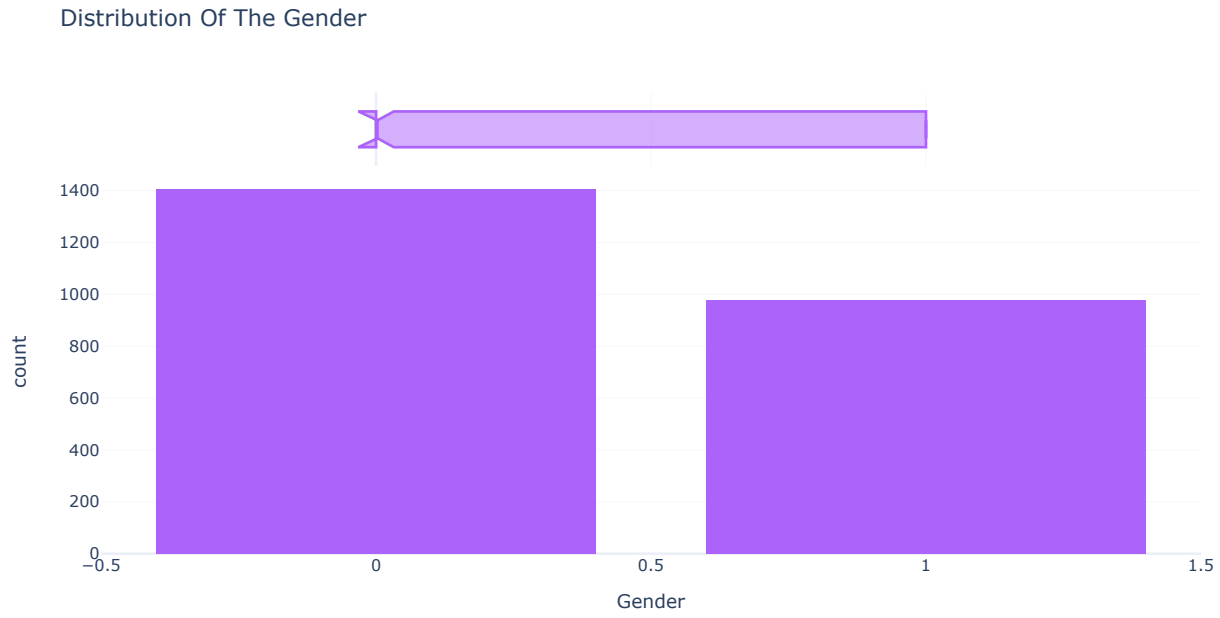
In [53]:

```
fig = px.histogram(df_1, x = 'Age' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The Age")
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



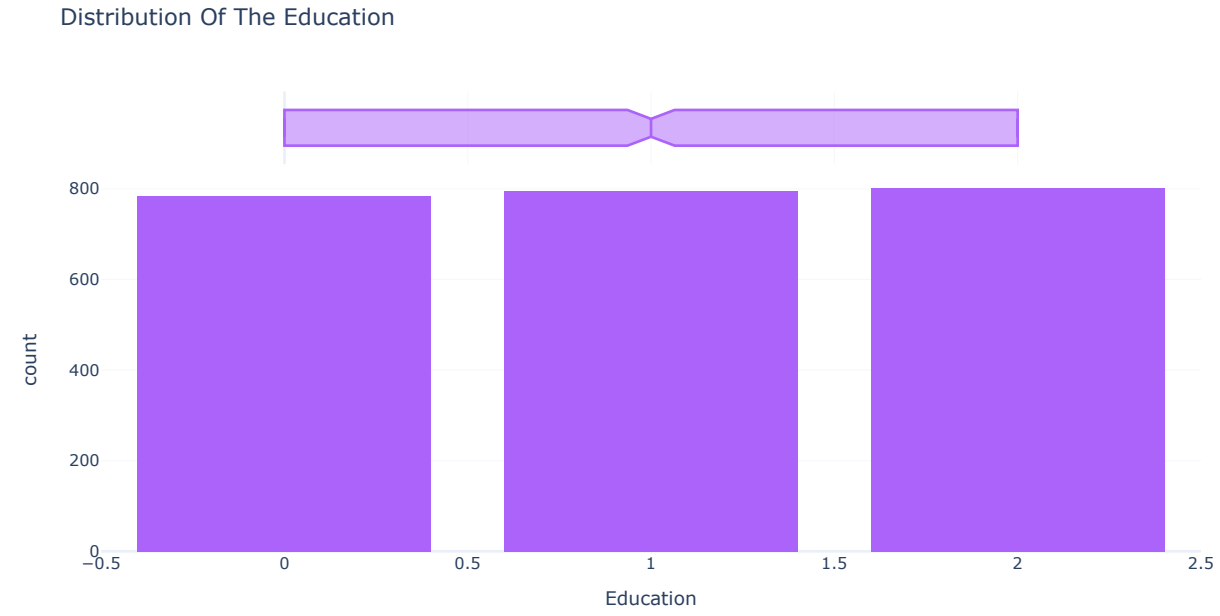
In [54]:

```
fig = px.histogram(df_1, x = 'Gender' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The Gender")
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



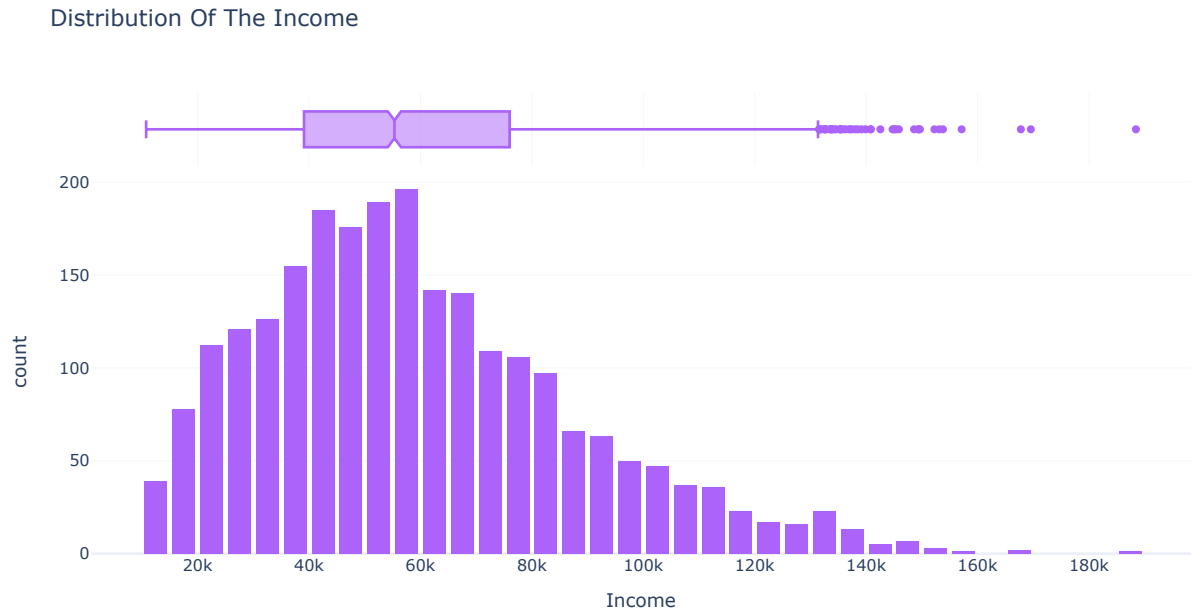
In [55]:

```
= px.histogram(df_1, x = 'Education' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The Education")
update_layout(bargap = 0.2 , template = 'plotly_white')
```



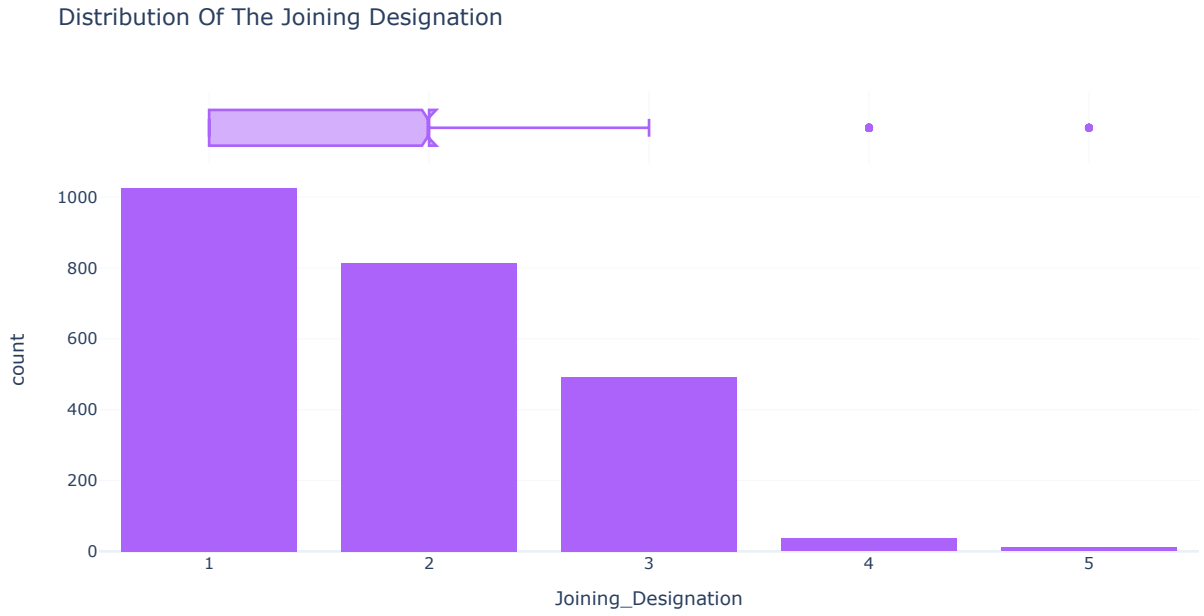
In [56]:

```
fig = px.histogram(df_1, x = 'Income' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The Income")
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



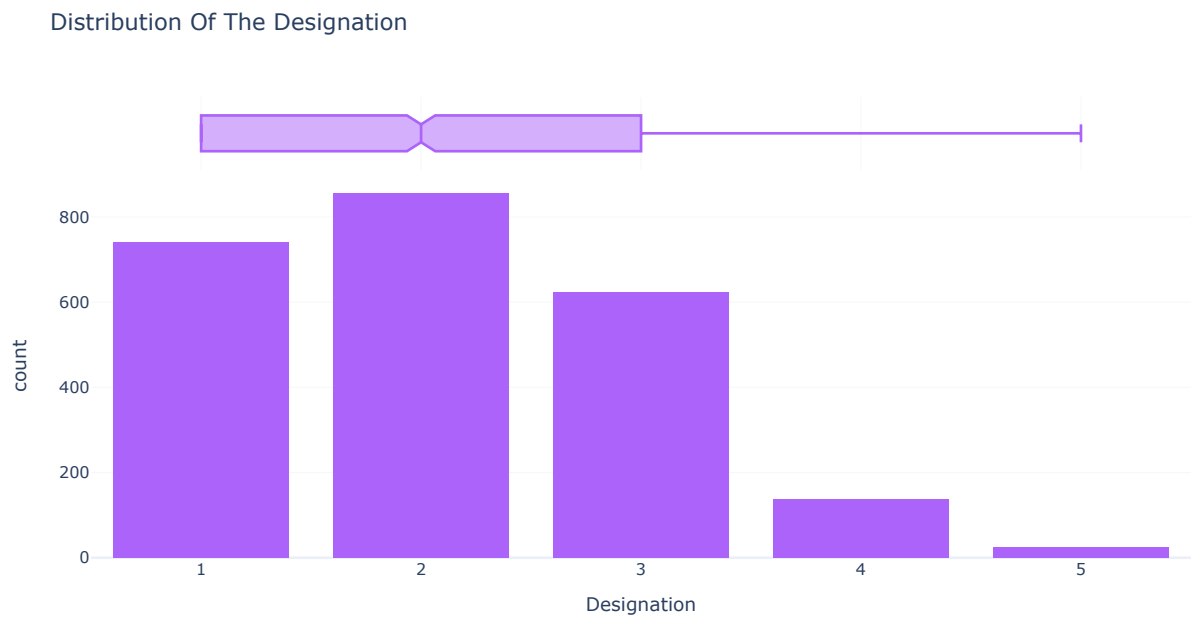
In [57]:

```
fig = px.histogram(df_1, x = 'Joining_Designation' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



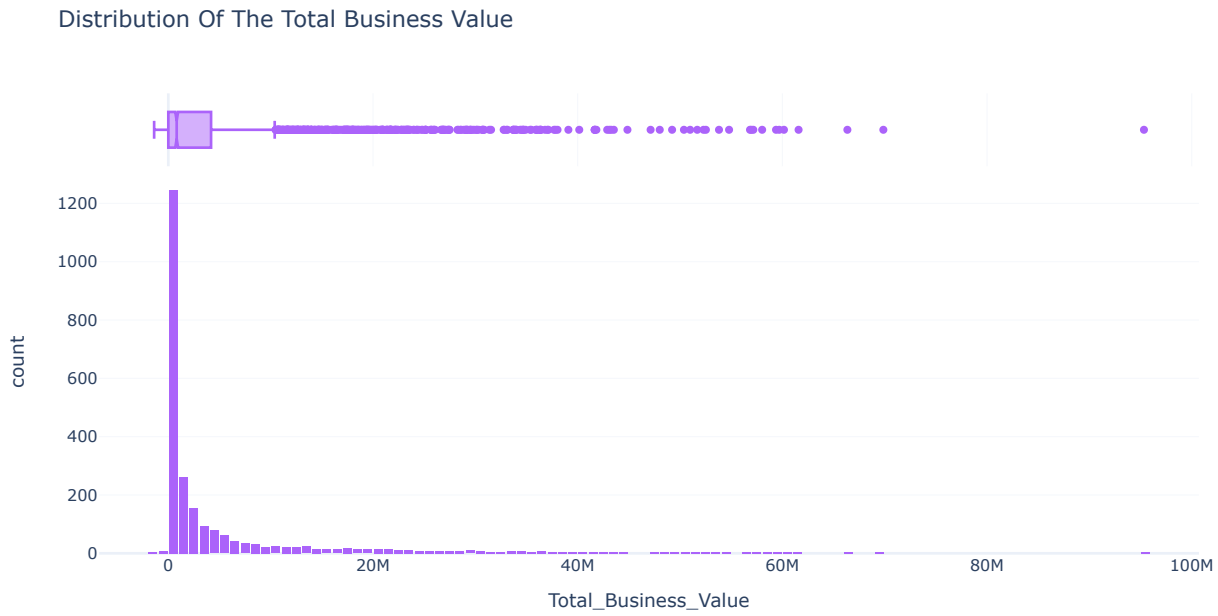
In [58]:

```
fig = px.histogram(df_1, x = 'Designation' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of The Designa
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



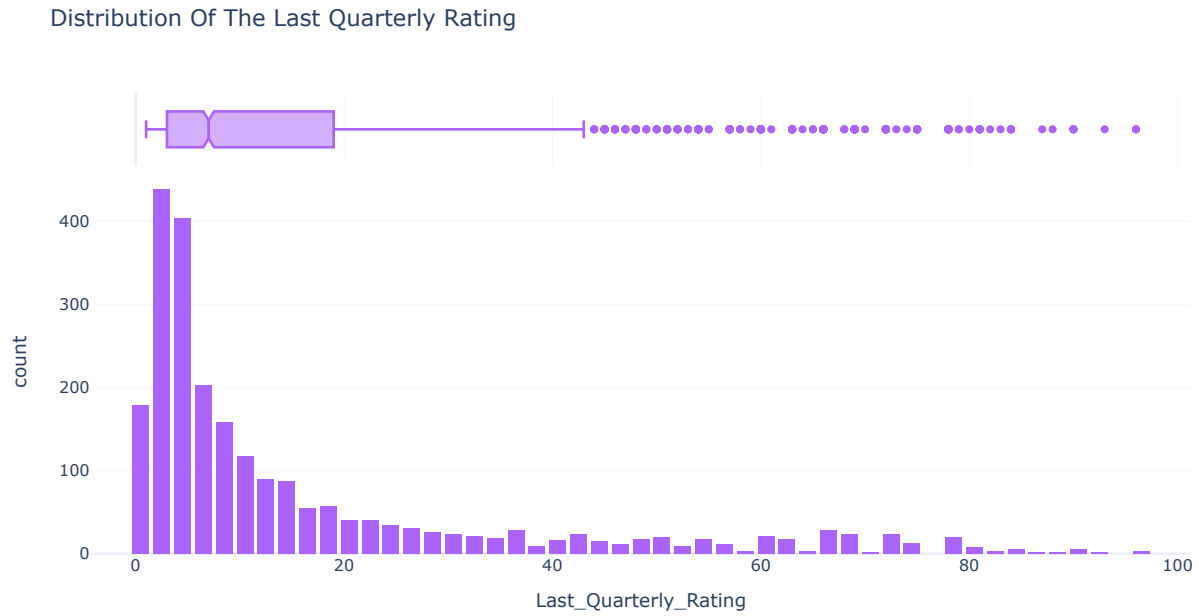
In [59]:

```
fig = px.histogram(df_1, x = 'Total_Business_Value' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of T
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



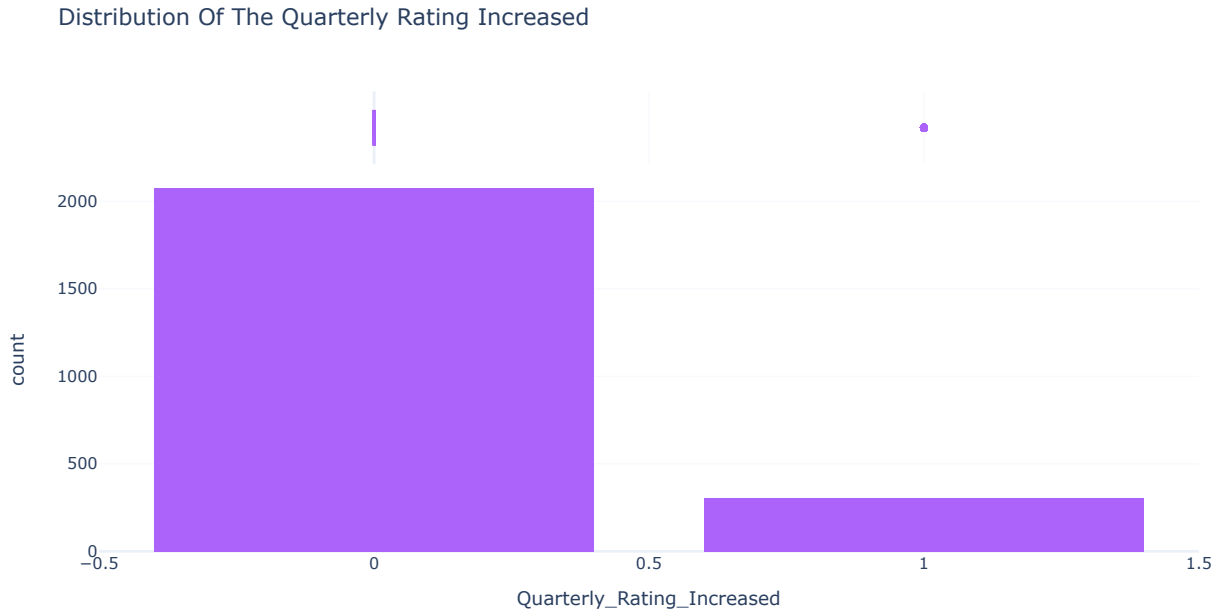
In [60]:

```
fig = px.histogram(df_1, x = 'Last_Quarterly_Rating' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distribution Of 
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



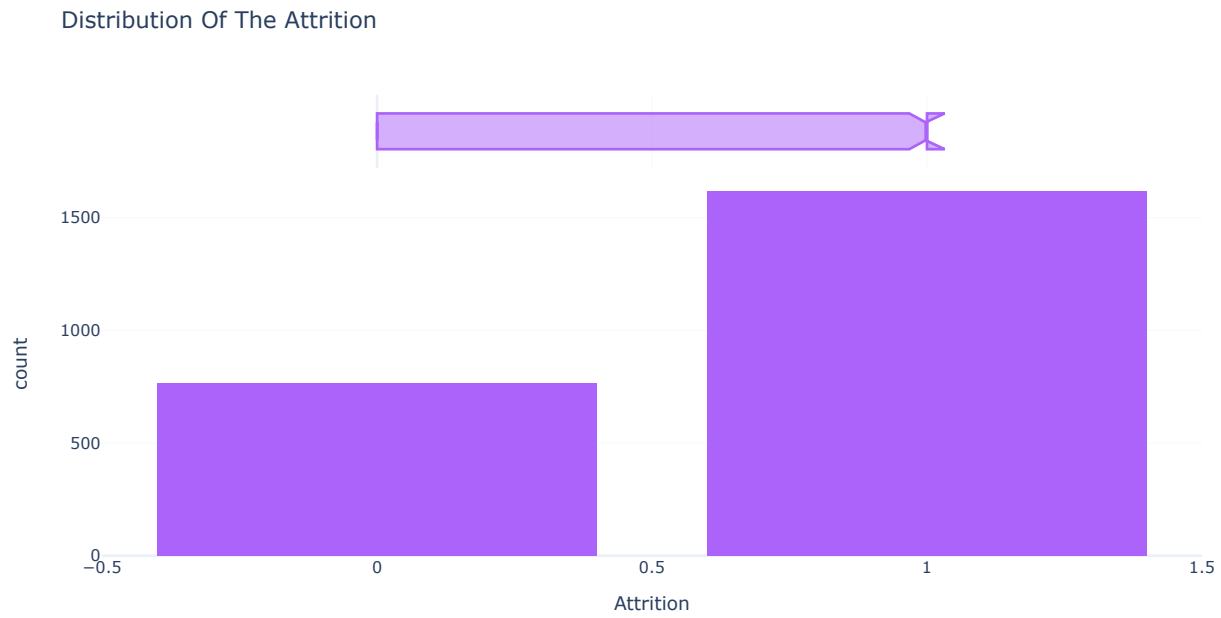
In [61]:

```
fig = px.histogram(df_1, x = 'Quarterly_Rating_Increased' , color_discrete_sequence = ['#AB63FA'] , marginal = "box" , title="Distributio
fig.update_layout(bargap = 0.2 , template = 'plotly_white')
```



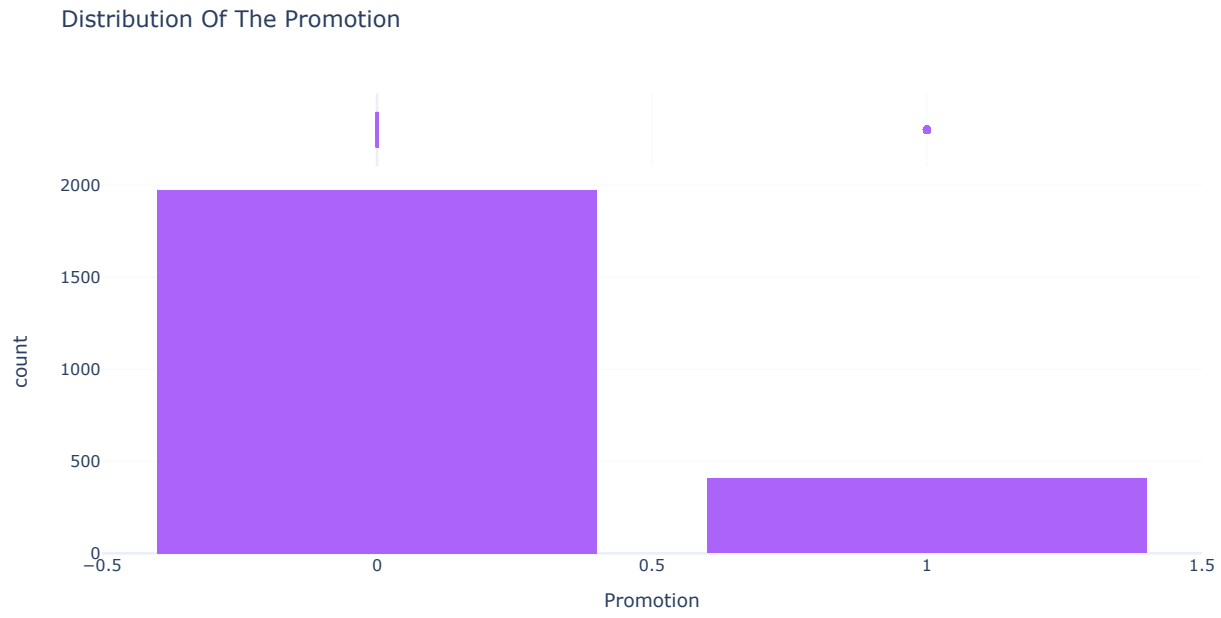
In [62]:

```
fig = px.histogram(df_1, x = 'Attrition',color_discrete_sequence=['#AB63FA'],marginal="box",title="Distribution Of The Attrition")
fig.update_layout(bargap=0.2,template='plotly_white')
```



In [63]:

```
fig = px.histogram(df_1, x = 'Promotion',color_discrete_sequence=['#AB63FA'],marginal="box",title="Distribution Of The Promotion")
fig.update_layout(bargap=0.2,template='plotly_white')
```



In [64]:

```
print('Maximum age of the students:',df['Age'].max())
print('Manimum age of the students:',df['Age'].min())
print('Average age of the students:',df['Age'].mean())
```

```
Maximum age of the students: 58.0
Manimum age of the students: 21.0
Average age of the students: 34.66387144053603
```

In [65]:

```
df_1.loc[(df['Age']>18)&(df['Age']<=25),'Emp_Age'] = 0
df_1.loc[(df['Age']>25)&(df['Age']<=35),'Emp_Age'] = 1
df_1.loc[(df['Age']>35)&(df['Age']<=45),'Emp_Age'] = 2
df_1.loc[(df['Age']>45)&(df['Age']<=55),'Emp_Age'] = 3
df_1.loc[df['Age']>55,'Emp_Age'] = 4

# converting 'Weight' from float to int
df_1['Emp_Age'] = df_1['Emp_Age'].astype(int)
```

In [66]:

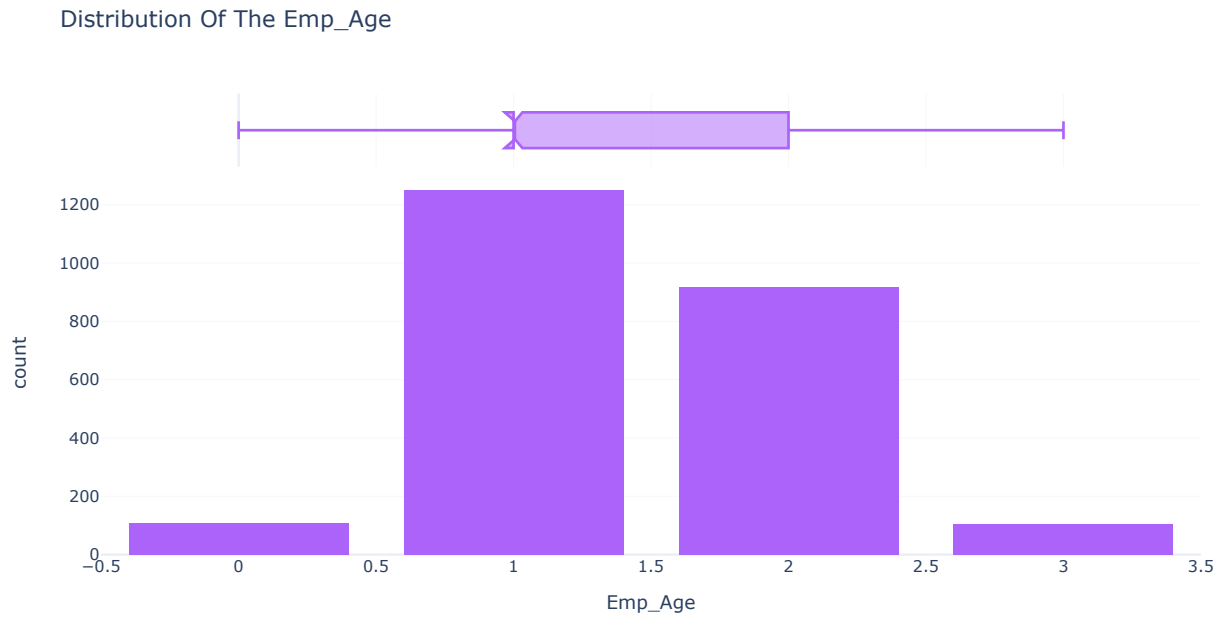
```
df_1['Emp_Age'].dtypes
```

Out[66]:

```
dtype('int32')
```

In [67]:

```
fig = px.histogram(df_1, x = 'Emp_Age',color_discrete_sequence=['#AB63FA'],title="Distribution Of The Emp_Age",marginal="box")
fig.update_layout(bargap=0.2,template='plotly_white')
```



Univariate Analysis on Categorical Features :

In [68]:

```
# list of Categorical variables.....
cat_features = [feature for feature in df_1.columns if ((df_1[feature].dtypes == 'O') & (df_1[feature].dtypes != 'datetime64[ns]'))]

print('Number of categorical variables: ', len(cat_features))
print('\n')
print('Categorical variables columns are: ',cat_features)
print('\n')
# visualise the numerical variables........
df_1[cat_features].head().style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-color": "white","font-size":"11
```

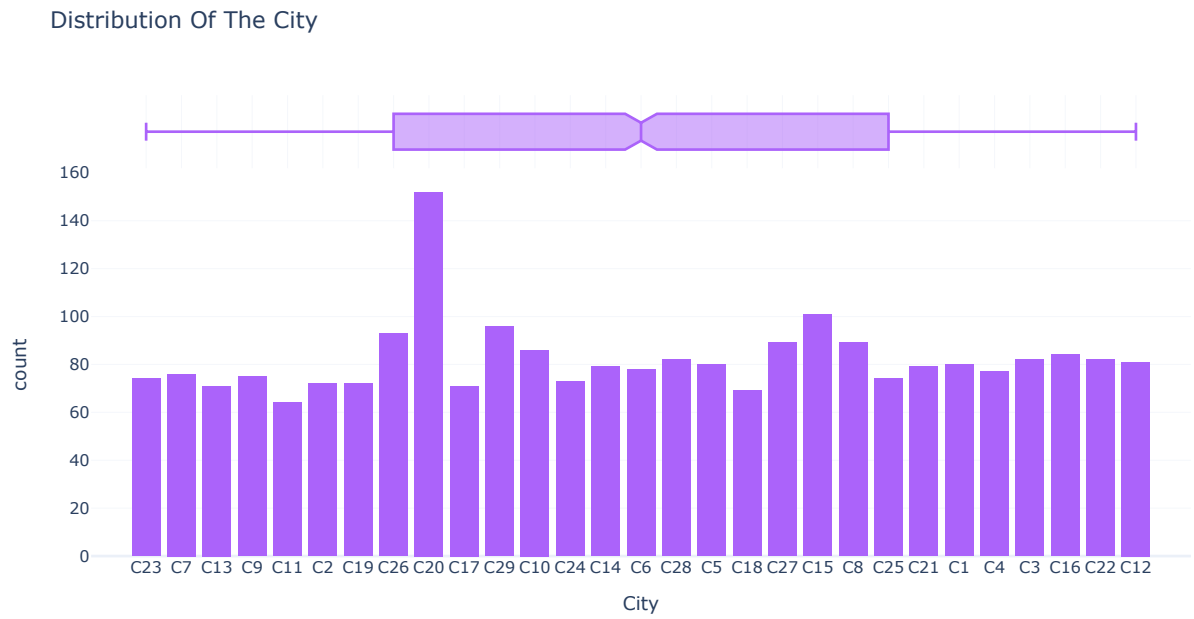
```
Number of categorical variables:  1


Categorical variables columns are:  ['City']
```

Out[68]:

	City
0	C23
1	C7
2	C13
3	C9
4	C11

In [69]:

```
fig = px.histogram(df_1 , x = 'City' , color_discrete_sequence = ['#AB63FA'] , title="Distribution Of The City" , marginal="box")
fig.update_layout( bargap=0.2 , template='plotly_white' )
```



Observations :

- Out of 2381 employees, 1404 employees are of the Male gender and 977 are females.
- Out of 2381 employees, 152 employees are from city C20 and 101 from city C15.
- Out of 2381 employees, 802 employees have their education as Masters and 795 have completed their Bachelors.
- Out of 2381 employees, 1026 joined with the designation as 1, 815 employees joined with the designation 2.
- Out of 2381 employees, 855 employees had their designation as 2 at the time of reporting.
- Out of 2381 employees, 1744 employees had their last quarterly rating as 1.
- Out of 2381 employees, the quarterly rating has not increased for 2076 employees.
- Around 6.4% employees are from city C20 and 4.2% from city C15.
- The proportion of the employees who have completed their Masters and Bachelors is approximately same.
- Around 43% of the employees joined with the designation 1.
- At the time of reporting, 34% of the employees had their designation as 2.
- Around 73% of the employees had their last quarterly rating as 1.

- The quarterly rating has not increased for around 87% employees.

Bivariate Analysis :

a) Driver Age vs Attrition :

In [70]:

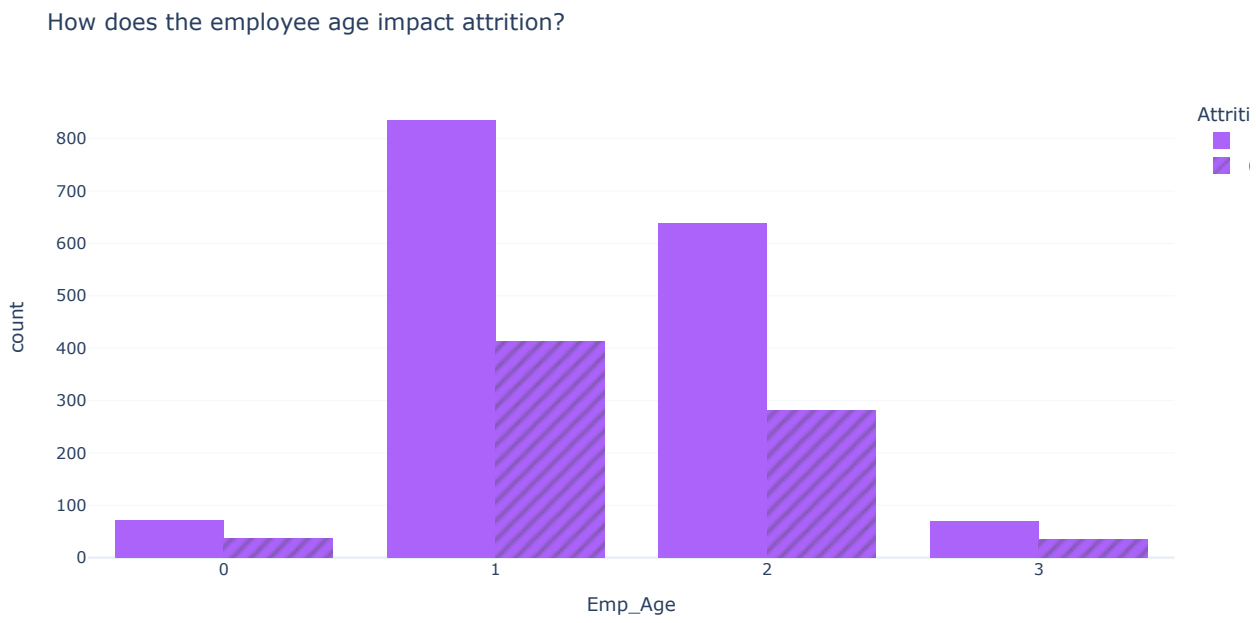
```
pd.crosstab(df_1.Emp_Age,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-colo
```

Out[70]:

Attrition	0	1	All
Emp_Age			
0	36	72	108
1	413	836	1249
2	281	638	919
3	35	70	105
All	765	1616	2381

In [71]:

```
fig = px.histogram(df_1, x="Emp_Age", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the employee age impact a
fig.update_layout(bargap=0.2,template='plotly_white')
```



Most of the Employees whose age is between 25 to 45 are leaving the organization.

b) Gender vs Attrition :

In [72]:

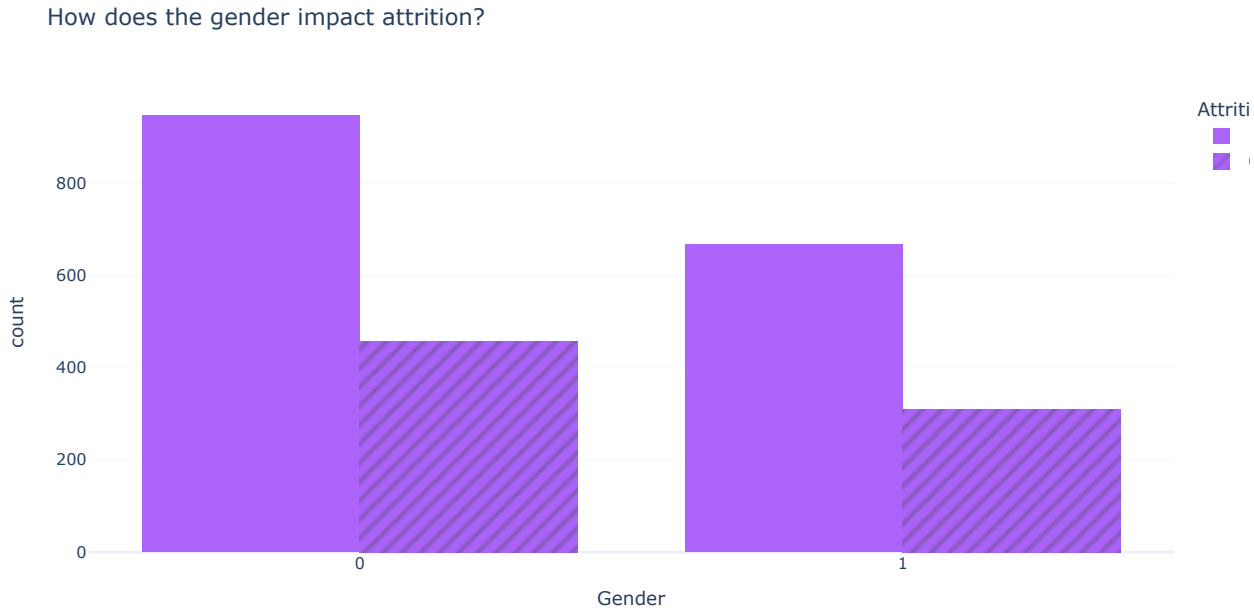
```
pd.crosstab(df_1.Gender,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-colo
```

Out[72]:

Attrition	0	1	All
Gender			
0.000000	456	948	1404
1.000000	309	668	977
All	765	1616	2381

In [73]:

```
fig = px.histogram(df_1, x= "Gender", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the gender impact attriti
fig.update_layout(bargap=0.2,template='plotly_white')
```



Most of the employees are male who are leaving the organization.

c) City vs Attrition :

In [74]:

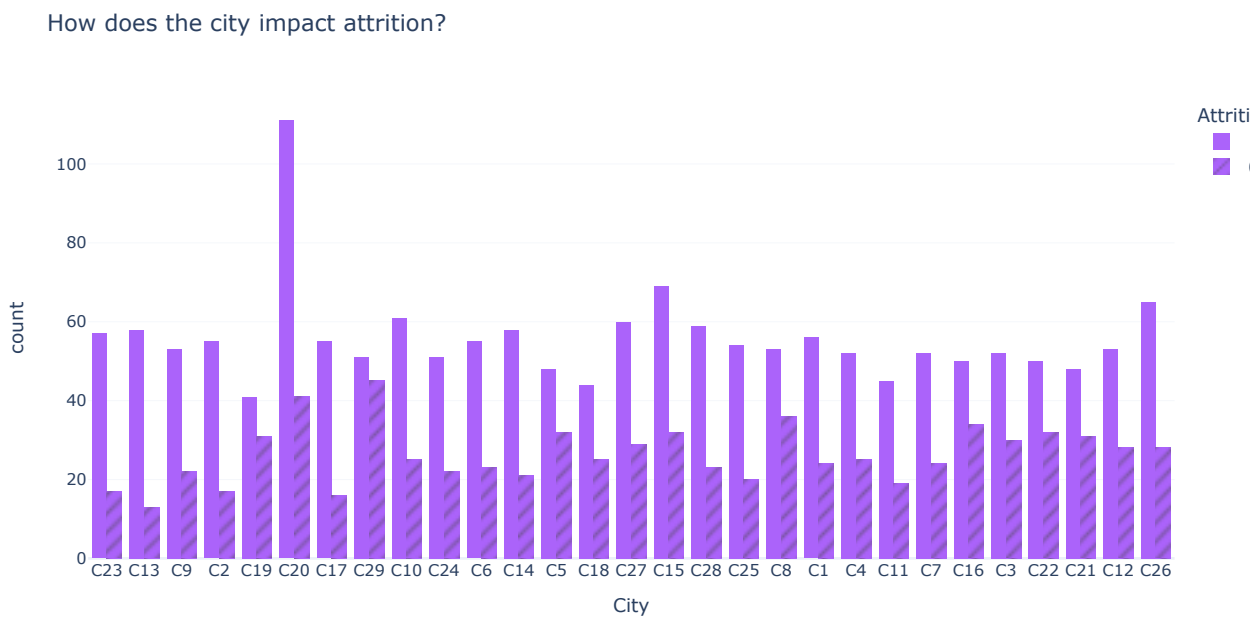
```
pd.crosstab(df_1.City,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-color"
```

Out[74]:

Attrition	0	1	All
City			
C1	24	56	80
C10	25	61	86
C11	19	45	64
C12	28	53	81
C13	13	58	71
C14	21	58	79
C15	32	69	101
C16	34	50	84
C17	16	55	71
C18	25	44	69
C19	31	41	72
C2	17	55	72
C20	41	111	152
C21	31	48	79
C22	32	50	82
C23	17	57	74
C24	22	51	73
C25	20	54	74
C26	28	65	93
C27	29	60	89
C28	23	59	82
C29	45	51	96
C3	30	52	82
C4	25	52	77
C5	32	48	80
C6	23	55	78
C7	24	52	76
C8	36	53	89
C9	22	53	75
All	765	1616	2381

In [75]:

```
fig = px.histogram(df_1, x="City", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the city impact attritior
fig.update_layout(bargap=0.2,template='plotly_white')
```



Most of the employees who are fom "C20" are leaving the organization.

d) Education vs Attrition :

In [76]:

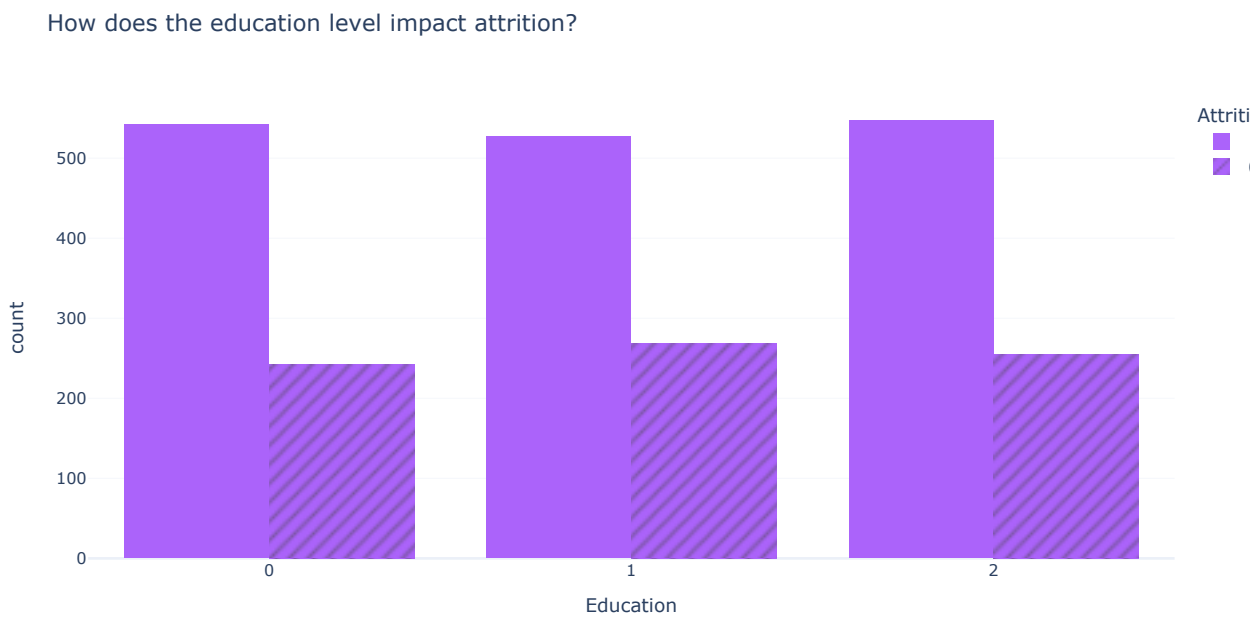
```
pd.crosstab(df_1.Education ,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-
```

Out[76]:

Attrition	0	1	All
Education			
0	242	542	784
1	268	527	795
2	255	547	802
All	765	1616	2381

In [77]:

```
fig = px.histogram(df_1, x="Education", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the education level impac
fig.update_layout(bargap=0.2,template='plotly_white')
```

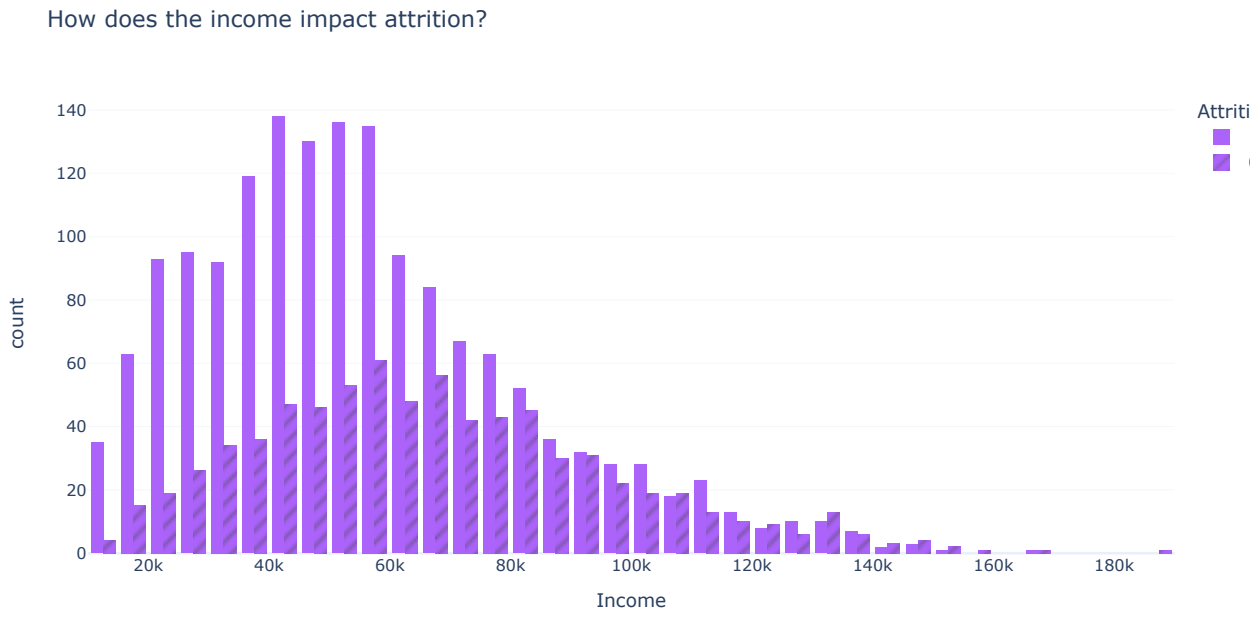


Looks like leaving organisation is not at all related to educational level.

e) Salary vs Attrition :

In [78]:

```
px.histogram(df_1, x="Income", color ="Attrition", pattern_shape="Attrition",
             template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title="How does the income impact attrition?")
)date_layout(bargap=0.2,template='plotly_white')
```



- Most of the employees whose salary is between 20k-80k are leaving the organization.
- Obviosly if the employee is earning good salary that why would he/she leave the job.

f) Promotion vs Attrition :

In [79]:

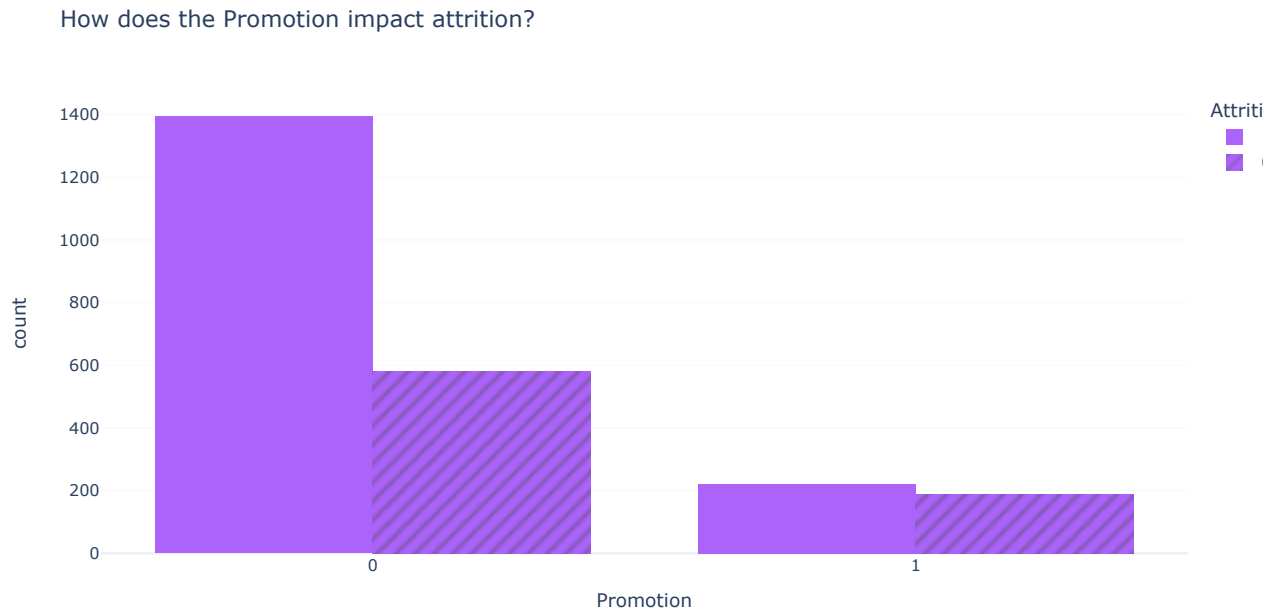
```
pd.crosstab(df_1['Promotion'] ,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color": "white", "borde
```

Out[79]:

Attrition	0	1	All
Promotion			
0	579	1395	1974
1	186	221	407
All	765	1616	2381

In [80]:

```
fig = px.histogram(df_1, x="Promotion", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the Promotion impact attr
fig.update_layout(bargap=0.2,template='plotly_white')
```



- Most of the employees are not getting promoted may be that is the reason of leaving the organization.
- Very few employees are getting acknowledged their work at the organization.

g) Total Business Value vs Attrition :

In [81]:

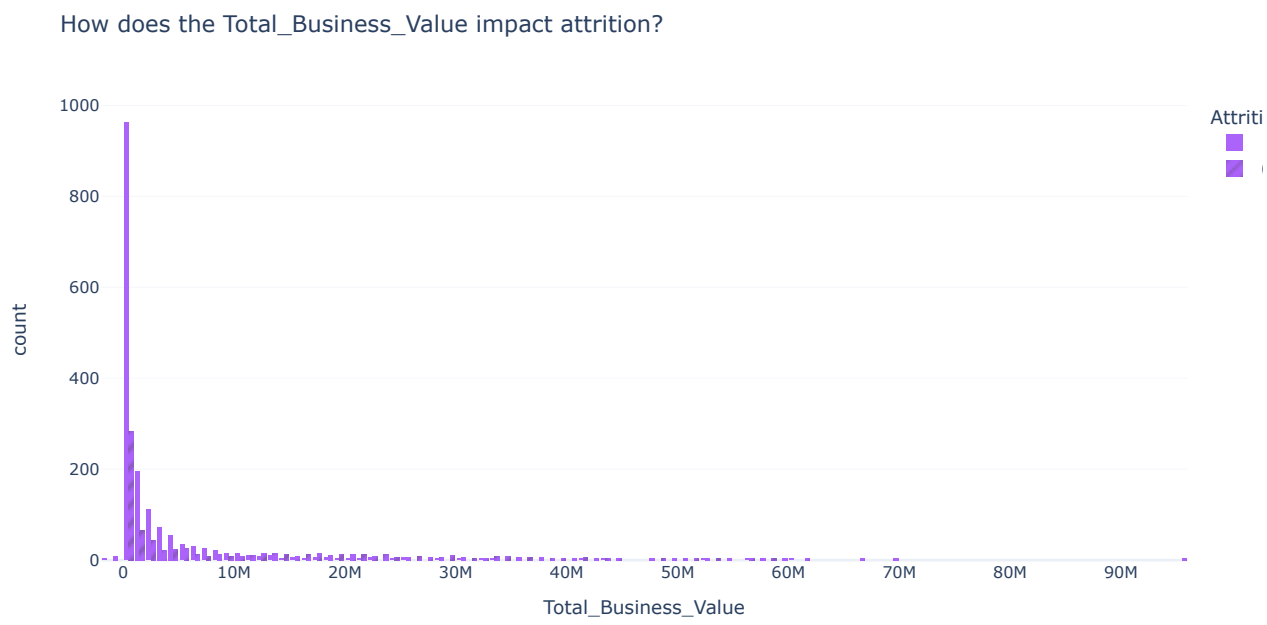
```
pd.crosstab(df_1['Total_Business_Value'] ,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color": "wh
```

Out[81]:

Attrition	0	1	All
Total_Business_Value			
-1385530	0	1	1
-645150	0	1	1
-500000	0	1	1
-439300	0	1	1
-411250	0	1	1
-134880	0	1	1
-101180	0	1	1

In [82]:

```
fig = px.histogram(df_1, x="Total_Business_Value", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the Total_Business_Value
fig.update_layout(bargap=0.2,template='plotly_white')
```



Most of the employees whose total business values is 5% are the highest no. of employees who are leaving the organization.

h) Quarterly Increased Rating vs Attrition :

In [83]:

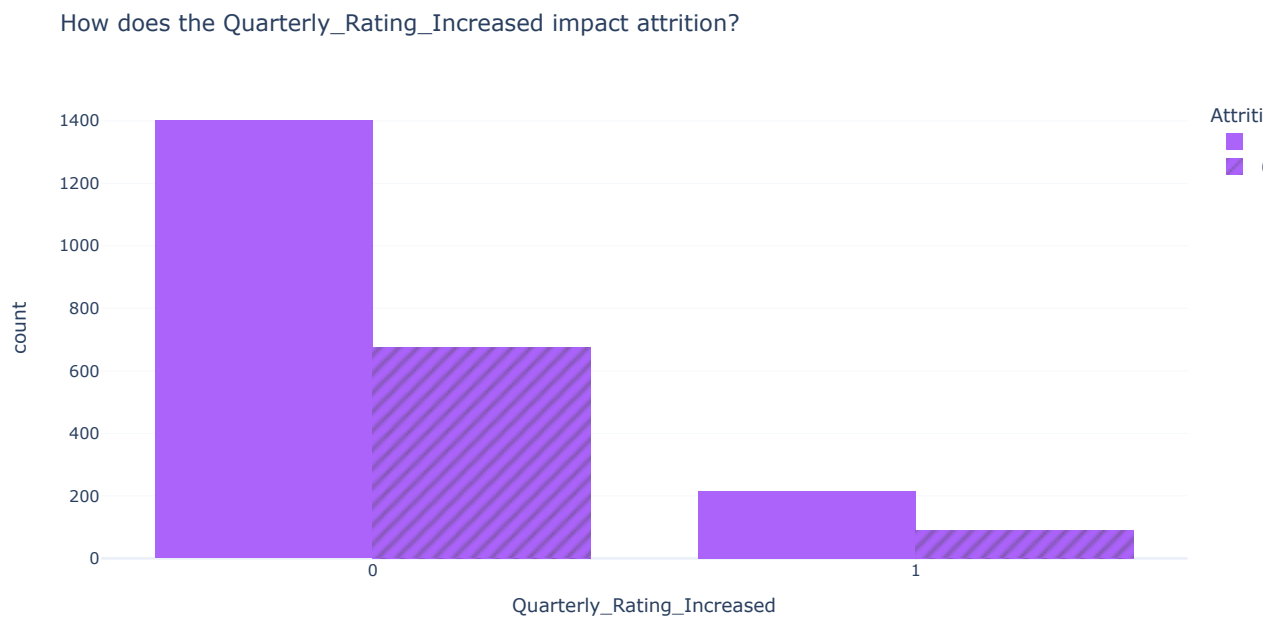
```
pd.crosstab(df_1['Quarterly_Rating_Increased'] ,df_1.Attrition,margins=True).style.set_properties(**{"background-color": "#AB63FA","color"
```

Out[83]:

Attrition	0	1	All
Quarterly_Rating_Increased			
0	675	1401	2076
1	90	215	305
All	765	1616	2381

In [84]:

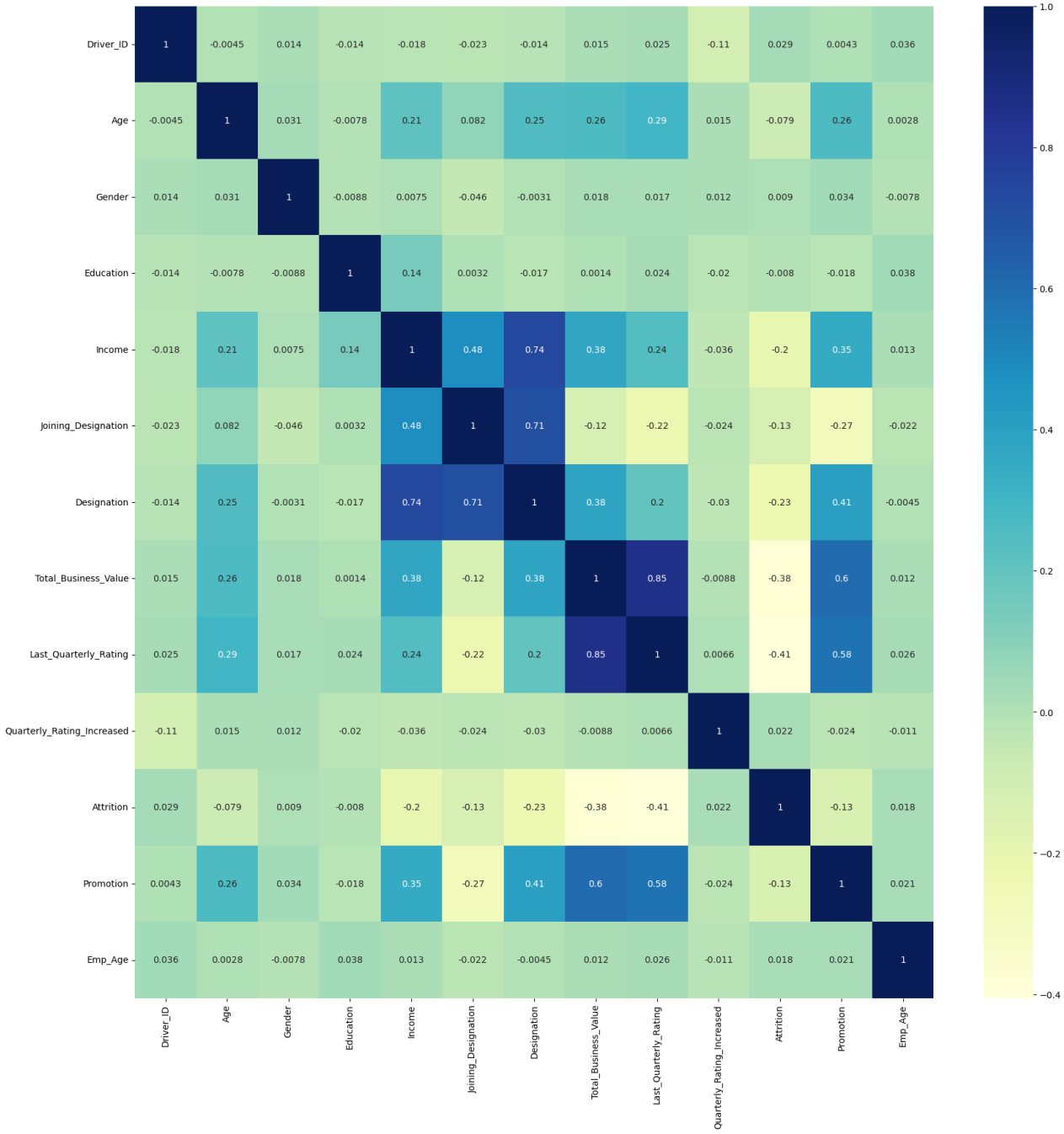
```
fig = px.histogram(df_1, x="Quarterly_Rating_Increased", color ="Attrition", pattern_shape="Attrition",
                   template='plotly_white', barmode='group',color_discrete_sequence=['#AB63FA'],title= "How does the Quarterly_Rating_Incr
fig.update_layout(bargap=0.2,template='plotly_white')
```



- Most of the employees are getting low rating may be could be the reason of leaving the organization.
- Very few employees are getting acknowledged their work at the organization and getting good rating.

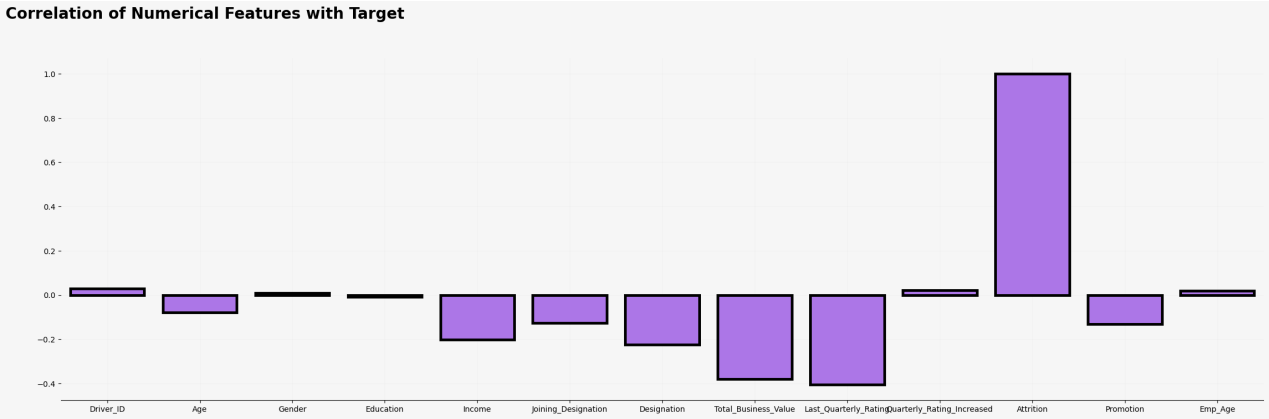
In [85]:

```
fig, ax = plt.subplots(figsize=(20,20))
sns.heatmap(df_1.corr(),cmap='YlGnBu',annot=True,annot_kws={'fontsize':10})
plt.show()
```



In [86]:

```
import matplotlib
background_color = "#f6f6f6"

fig = plt.figure(figsize=(28,8), facecolor=background_color)
gs = fig.add_gridspec(1, 1)
ax0 = fig.add_subplot(gs[0, 0])
colors = ["#2f5586", "#f6f5f5","#2f5586"]
colormap = matplotlib.colors.LinearSegmentedColormap.from_list("", colors)

ax0.set_facecolor(background_color)
ax0.text(-1.1, 1.25, 'Correlation of Numerical Features with Target', fontsize=20, fontweight='bold')

chart_df = pd.DataFrame(df_1.corrwith(df_1['Attrition']))
chart_df.columns = ['corr']
sns.barplot(x=chart_df.index, y=chart_df['corr'], ax=ax0, color='#AB63FA', zorder=3, edgecolor='black', linewidth=3.5)
ax0.grid(which='major', axis='x', zorder=0, color='#EEEEEE', linewidth=0.4)
ax0.grid(which='major', axis='y', zorder=0, color='#EEEEEE', linewidth=0.4)
ax0.set_ylabel('')

for s in ["top","right", 'left']:
    ax0.spines[s].set_visible(False)

plt.show()
```



After performing correlation we can say that Emp_Id , Quarterly_Rating_Increased , Salary_Increased are highly correlated.

In [87]:

```
from sklearn.model_selection import train_test_split
```

In [88]:

```
cols = ['Driver_ID', 'Age','Gender', 'Education' , 'Income',
       'Joining_Designation', 'Designation', 'Total_Business_Value',
       'Last_Quarterly_Rating' , 'Quarterly_Rating_Increased' , 'Promotion' , 'Emp_Age']
X = df_1[cols]
y = df_1['Attrition']

X_train , X_test , y_train , y_test = train_test_split(X , y , random_state=42 , test_size=0.25 , shuffle=True )

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1785, 12)
(596, 12)
(1785,)
(596,)
```

In [89]:

```
from imblearn.over_sampling import SMOTE
from collections import Counter
```

In [90]:

```
sm = SMOTE()
x_sm,y_sm = sm.fit_resample(X_train,y_train)
print('Resampled data has {}'.format(Counter(y_sm)))
```

```
Resampled data has Counter({0: 1213, 1: 1213})
```

In [91]:

```
x_sm.head()
```

Out[91]:

	Driver_ID	Age	Gender	Education	Income	Joining_Designation	Designation	Total_Business_Value	Last_Quarterly_Rating	Quarterly_Rating_Increased
0	1632	36.0	1.0	1	81876	2	2	0	1	0
1	2649	40.0	0.0	2	69098	2	2	8924290	37	0
2	1386	27.0	1.0	2	25458	1	1	1164500	12	0
3	2558	37.0	0.0	1	29962	1	1	5818480	35	0
4	137	42.0	0.0	0	33459	2	2	6570070	22	0

In [92]:

```
y_sm.head()
```

Out[92]:

```
0    0
1    1
2    1
3    1
4    0
Name: Attrition, dtype: int64
```

In [93]:

```
df_1 = pd.get_dummies(data = df_1 , columns = ['City'] , drop_first=True)
df_1.head().style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-color": "white","font-size":"11.5pt",'width':
```

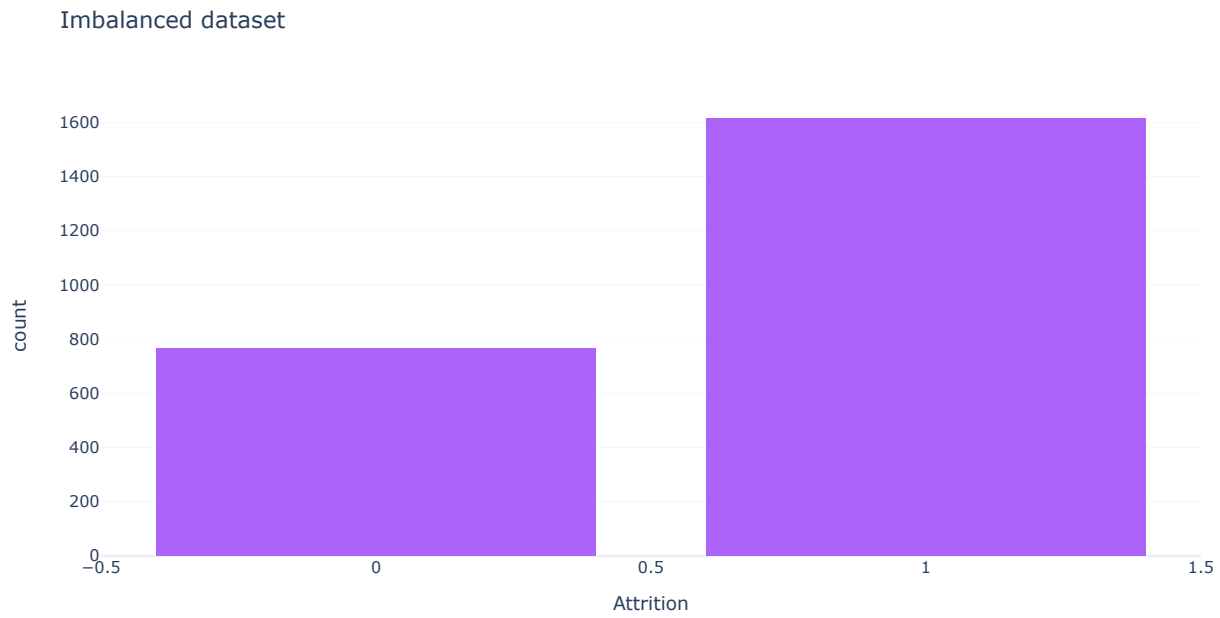
Out[93]:

	Driver_ID	Age	Gender	Education	Income	Joining_Designation	Designation	Total_Business_Value	Last_Quarterly_Rating	Quarterly_Rati
0	1	28.000000	0.000000	2	57387	1	1	1715580	6	
1	2	31.000000	0.000000	2	67016	2	2	0	2	
2	4	43.000000	0.000000	2	65603	2	2	350000	5	
3	5	29.000000	0.000000	0	46368	1	1	120360	3	
4	6	31.000000	1.000000	1	78728	3	3	1265000	8	

In [94]:

```
print("Value_count:",df_1['Attrition'].value_counts())
print("Shape:",df_1.shape)
fig = px.histogram(df_1, x = 'Attrition',color_discrete_sequence=['#AB63FA'],title="Imbalanced dataset")
fig.update_layout(bargap=0.2,template='plotly_white')
```

```
Value_count: 1    1616
0     765
Name: Attrition, dtype: int64
Shape: (2381, 41)
```



In [95]:

```
#Using oversampling technique i am trying to balance the dataset...

from sklearn.utils import resample

#Separate majority and minority classes
df_majority = df_1[df_1.Attrition==1]
df_minority = df_1[df_1.Attrition==0]

# Upsample minority class
df_minority_upsampled = resample(df_minority, replace=True,n_samples=1616)      # sample with replacement

# Combine majority class with upsampled minority class
df_upsampled = pd.concat([df_majority, df_minority_upsampled])

print(df_upsampled['Attrition'].value_counts())
X=df_upsampled['Attrition'].value_counts()
```

```
1    1616
0    1616
Name: Attrition, dtype: int64
```

In [96]:

```
#Rearrangement of the columns
df_1 = df_1[['Driver_ID','Emp_Age','City_C10', 'City_C11',
       'City_C12', 'City_C13', 'City_C14', 'City_C15', 'City_C16', 'City_C17',
       'City_C18', 'City_C19', 'City_C2', 'City_C20', 'City_C21', 'City_C22',
       'City_C23', 'City_C24', 'City_C25', 'City_C26', 'City_C27', 'City_C28',
       'City_C29', 'City_C3', 'City_C4', 'City_C5', 'City_C6', 'City_C7',
       'City_C8', 'City_C9','Income', 'Joining_Designation', 'Designation',
       'Total_Business_Value', 'Last_Quarterly_Rating',
       'Quarterly_Rating_Increased','Promotion', 'Attrition']]

df_1.head(5).style.set_properties(**{"background-color": "#AB63FA","color": "white", "border-color": "white","font-size":"11.5pt",'width'
```

Out[96]:

	Driver_ID	Emp_Age	City_C10	City_C11	City_C12	City_C13	City_C14	City_C15	City_C16	City_C17	City_C18	City_C19	City_C2	City_C20	City_C2
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	4	1	0	0	0	1	0	0	0	0	0	0	0	0	0
3	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	6	1	0	1	0	0	0	0	0	0	0	0	0	0	0

Standardization :

In [97]:

```
from sklearn.preprocessing import MinMaxScaler
#MinMaxScaler
scaler = MinMaxScaler()

scaler.fit(x_sm)
df_scaled = scaler.transform(x_sm)
df_scaled
```

Out[97]:

```
array([[0.58521708, 0.44117647, 1.        , ..., 0.        , 0.        ,
        0.33333333],
       [0.95012558, 0.55882353, 0.        , ..., 0.        , 0.        ,
        0.33333333],
       [0.49695013, 0.17647059, 1.        , ..., 0.        , 0.        ,
        0.66666667],
       ...,
       [0.30426982, 0.15025527, 0.61141509, ..., 0.        , 0.        ,
        0.33333333],
       [0.64621457, 0.49362676, 1.        , ..., 0.        , 1.        ,
        0.        ],
       [0.38141371, 0.65487043, 1.        , ..., 0.        , 1.        ,
        0.33333333]])
```

In [98]:

```
df_2 = df_1.copy()
df_1.drop(columns = ["Total_Business_Value" , "Income" , "Driver_ID"] , inplace = True)
```

Model Creation using Bagging and Boosting :

In [99]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold, learning_curve ,KFold
from sklearn.metrics import roc_curve,accuracy_score,f1_score,roc_curve,confusion_matrix,roc_auc_score,plot_confusion_matrix
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,ConfusionMatrixDisplay
```

Random Forest Model (Bagging):

In [100]:

```
rf = RandomForestClassifier(max_depth= 7, n_estimators= 100)
rf.fit(X_train, y_train)
```

Out[100]:

```
RandomForestClassifier(max_depth=7)
```

In [101]:

```
rf_y_preds = rf.predict(X_test)

print(classification_report(y_test , rf_y_preds))
```

```
              precision    recall  f1-score   support

           0       0.82      0.36      0.50       193
           1       0.76      0.96      0.85       403

    accuracy                           0.77       596
   macro avg       0.79      0.66      0.68       596
weighted avg       0.78      0.77      0.74       596
```

In [102]:

```
print("The f1-score = ",f1_score(y_test,rf_y_preds))
```

```
The f1-score =  0.849015317286652
```

In [103]:

```
conf_matrix_rf = confusion_matrix(y_test,rf_y_preds)
conf_matrix_rf
```

Out[103]:

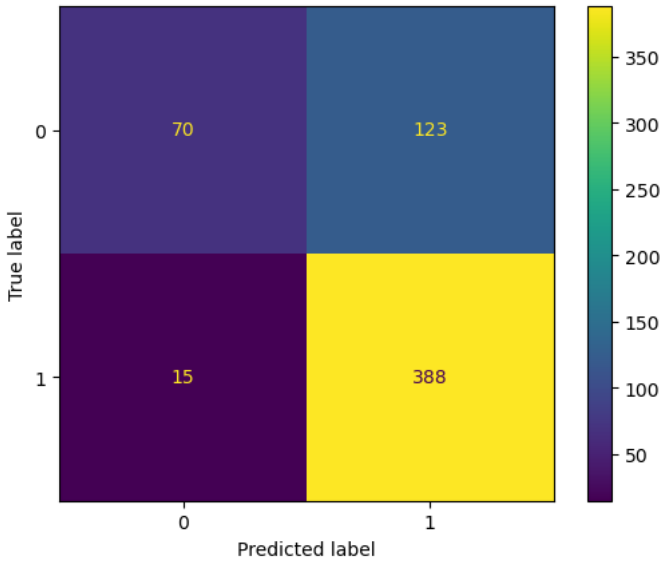
```
array([[ 70, 123],
       [ 15, 388]], dtype=int64)
```

In [104]:

```
ConfusionMatrixDisplay(conf_matrix_rf).plot()
```

Out[104]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x18817111f40>
```



In [105]:

```
np.diag(conf_matrix_rf).sum() / conf_matrix_rf.sum()*100
```

Out[105]:

```
76.84563758389261
```

Gradient Boosting Model :

In [106]:

```
gb = GradientBoostingClassifier()

parameters = {
    'loss': ['deviance', 'exponential'],
    'learning_rate': [0.001, 0.1, 1, 10],
    'n_estimators': [100, 150, 180, 200]
}

grid_search = GridSearchCV(gb, parameters, cv = 5, n_jobs = -1, verbose = 1)
grid_search.fit(X_train, y_train)

print(f'Best parameters are : {grid_search.best_params_}')
print(f'The score is : {grid_search.best_score_}')
```

```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best parameters are : {'learning_rate': 0.1, 'loss': 'exponential', 'n_estimators': 100}
The score is : 0.7697478991596638
```

Since Accuracy for Random Forest is more than that of Gradient Boosting, we can prefer Random Forest Technique.

ROC - AUC Curve :

In [107]:

```
from sklearn.metrics import RocCurveDisplay
```

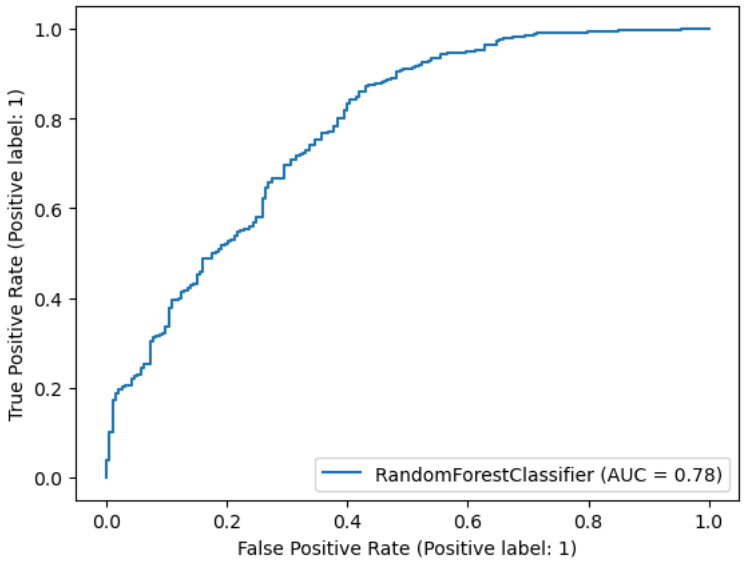
For Random Forest :

In [108]:

```
RocCurveDisplay.from_estimator(rf, X_test, y_test)
```

Out[108]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x188171d3070>
```

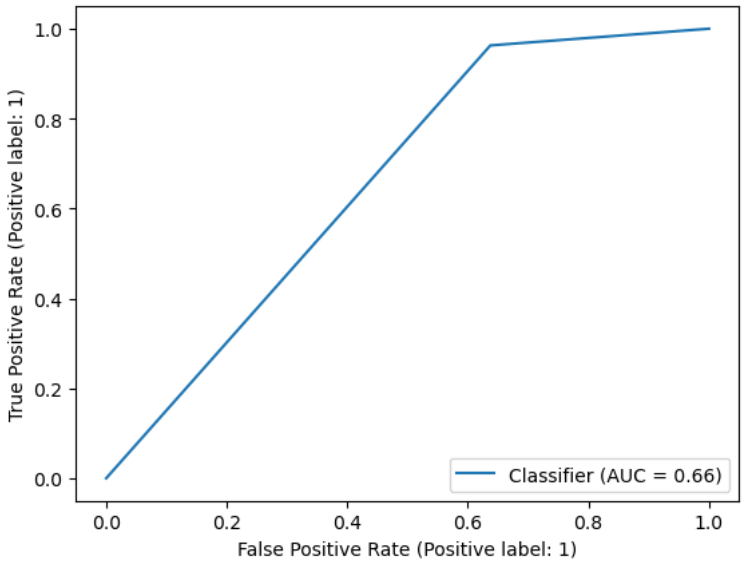


In [109]:

```
RocCurveDisplay.from_predictions(y_test, rf_y_preds)
```

Out[109]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x188173252b0>
```



Insights :

- Around 6.4% employees are from city C20 and 4.2% from city C15.
- The proportion of the employees who have completed their Masters and Bachelors is approximately same.
- Around 43% of the employees joined with the designation 1.
- At the time of reporting, 34% of the employees had their designation as 2.
- Around 73% of the employees had their last quarterly rating as 1.
- The quarterly rating has not increased for around 87% employees.
- Most of the Employees whose age is between 25 to 45 are leaving the organization.
- Most of the employees are male who are leaving the organization.
- Most of the employees who are fom "C20" are leaving the organization.
- Most of the employees whose salary is between 20k-80k are leaving the organization.
- Most of the employees are not getting promoted may be that is the reason of leaving the organization.
- Very few employees are getting acknowledged their work at the organization.
- Most of the employees whose total business values is 5% are the highest no. of employees who are leaving the organization.
- Most of the employees are getting low rating may be could be the reason of leaving the organization.
- Very few employees are getting acknowledged their work at the organization and getting good rating.
- F1 Score => 85.5 and Accuracy (for Random Forest) => 76.84. And Score (for Boosting) => 76.97
- AOC-RUC for Random Forest from Estimators => 0.78 , and from Predictions => 0.66.

Recommendations :

- Months Worked is one of the important feature, also from the graphical analysis its evident that the atleast 75 percentile of drivers that have left the organization have worked for 30 months or less. So if policies and arrangements can make sure that drivers stay upto 30-36 months after joining then its likely that they will be part of the organization for longer period of time. Also while hiring if we have an option of choosing between two candidates then its no brainer to pick up the candidate having higher average tenure at past organizations (given that this data is available).
- Average Monthly Income is also one of the important feature, hence good compensation model could ensure longer stay of the drivers, or else having some kind of bond for lower compensation driver could ensure their stay.
- Quarterly Rating and Business Value were highly correlated and hence developing a metric to monitor the drop of these values beyond a certain threshold for a particular driver could mean that there is some issue with the particular driver and hence it could be further discussed and any help needed could be extended to the extent possible, and this could help in decreasing attrition.
- City and Age also appears as important features. City for example could mean that bigger the city, better the opportunities from various ride business, again some sort of better policies could help decrease the attrition.

In [ ]:

```

```