Importing Libraries

```
import warnings
warnings.filterwarnings('ignore')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/con
```

Loading Dataset

```
pd.set_option('display.max_columns',None)
burnoutDf=pd.read_csv('/content/drive/MyDrive/employee_burnout_analysis.csv')
burnoutDf
```

	Employee ID	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	N Fa
0	fffe32003000360033003200	30-09-2008	Female	Service	No	2	3.0	
1	fffe3700360033003500	30-11-2008	Male	Service	Yes	1	2.0	
2	fffe31003300320037003900	10-03-2008	Female	Product	Yes	2	NaN	
3	fffe32003400380032003900	03-11-2008	Male	Service	Yes	1	1.0	
4	fffe31003900340031003600	24-07-2008	Female	Service	No	3	7.0	
...	...	...	...	...	...	...	...	...
22745	fffe31003500370039003100	30-12-2008	Female	Service	No	1	3.0	
22746	fffe33003000350031003800	19-01-2008	Female	Product	Yes	3	6.0	
22747	fffe390032003000	05-11-2008	Male	Service	Yes	3	7.0	
22748	fffe33003300320036003900	10-01-2008	Female	Service	No	2	5.0	
22749	fffe3400350031003800	06-01-2008	Male	Product	No	3	6.0	

22750 rows × 9 columns

Next steps: Generate code with burnoutDf | View recommended plots | New interactive sheet

```
#convert into dateTime dataType
burnoutDf["Date of Joining"]= pd.to_datetime(burnoutDf["Date of Joining"])


# give the number of rows and columns
burnoutDf.shape
```

```
(22750, 9)
```

```
# general information
burnoutDf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22750 entries, 0 to 22749
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   Employee ID          22750 non-null  object        
 1   Date of Joining      22750 non-null  datetime64[ns]
 2   Gender               22750 non-null  object        
 3   Company Type         22750 non-null  object        
 4   WFH Setup Available  22750 non-null  object        
 5   Designation          22750 non-null  int64         
 6   Resource Allocation  21369 non-null  float64       
```

```
 7   Mental Fatigue Score  20633 non-null  float64
 8   Burn Rate             21626 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(4)
memory usage: 1.6+ MB
```

```
#show top 5 rows
burnoutDf.head()
```

	Employee ID	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Menta Fatigu Scor
0	fffe32003000360033003200	2008-09-30	Female	Service	No	2	3.0	3
1	fffe3700360033003500	2008-11-30	Male	Service	Yes	1	2.0	5
2	fffe31003300320037003900	2008-03-10	Female	Product	Yes	2	NaN	5
3	fffe32003400380032003900	2008-11-03	Male	Service	Yes	1	1.0	2
4	fffe31003900340031003600	2008-07-24	Female	Service	No	3	7.0	6

Next steps: Generate code with burnoutDf | View recommended plots | New interactive sheet

```
#extract all columns of the dataset
burnoutDf.columns
```

```
Index(['Employee ID', 'Date of Joining', 'Gender', 'Company Type',
       'WFH Setup Available', 'Designation', 'Resource Allocation',
       'Mental Fatigue Score', 'Burn Rate'],
      dtype='object')
```

```
#check for null values
burnoutDf.isna().sum()
```

	0
Employee ID	0
Date of Joining	0
Gender	0
Company Type	0
WFH Setup Available	0
Designation	0
Resource Allocation	1381
Mental Fatigue Score	2117
Burn Rate	1124

```
#check the duplicates values
burnoutDf.duplicated().sum()
```

```
0
```

```
#calculate the mean,std,min,max,count of every attribute
burnoutDf.describe()
```

	Date of Joining	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
count	22750	22750.000000	21369.000000	20633.000000	21626.000000
mean	2008-07-01 09:28:05.274725120	2.178725	4.481398	5.728188	0.452005
min	2008-01-01 00:00:00	0.000000	1.000000	0.000000	0.000000
25%	2008-04-01 00:00:00	1.000000	3.000000	4.600000	0.310000
50%	2008-07-02 00:00:00	2.000000	4.000000	5.900000	0.450000
75%	2008-09-30 00:00:00	3.000000	6.000000	7.100000	0.590000
max	2008-12-31 00:00:00	5.000000	10.000000	10.000000	1.000000
std	NaN	1.135145	2.047211	1.920830	0.198226

```
#show the unique columns
for i, col in enumerate(burnoutDf.columns):
  print(f"\n\n{burnoutDf[col].unique()}")
  print(f"\n{burnoutDf[col].value_counts()}\n\n")
```

```
[0.16 0.36 0.49 0.2  0.52 0.29 0.62 0.33 0.56 0.67 0.5  0.12 0.4  0.51
 0.32 0.39 0.59 0.22 0.68 0.57 0.47 0.46 0.61 0.91 0.44 0.6  0.45 0.19
 0.31 0.81 0.42 0.53  nan 0.94 0.37 0.65 0.38 0.15 0.26 0.28 0.71 0.8
 0.63 0.79 0.72 0.34 0.27 0.66 0.04 0.05 0.11 0.41 0.76 0.43 0.85 0.35
 0.   0.55 0.48 0.7  0.18 0.23 0.25 0.75 0.1  0.73 0.58 0.88 0.77 0.3
 0.06 0.03 0.69 0.24 0.74 0.86 0.92 0.78 0.21 0.98 0.02 0.82 0.93 0.83
 0.87 0.64 0.54 0.17 1.   0.08 0.09 0.14 0.13 0.07 0.84 0.99 0.01 0.97
 0.95 0.9  0.96 0.89]

Burn Rate
0.47    475
0.43    444
0.41    434
0.45    431
0.50    428
       ...
0.98     18
0.97     17
0.95     17
0.96     13
0.99      8
Name: count, Length: 101, dtype: int64
```

```
#drop irrelevant column
burnoutDf= burnoutDf.drop(['Employee ID'],axis=1)
```

```
#check the skewness of the attributes
intFloatburnoutDf = burnoutDf.select_dtypes([int, float])

for i, col in enumerate(intFloatburnoutDf.columns):
    skew_value = intFloatburnoutDf[col].skew()
    if skew_value >= 0.1:
        print("\n", col, "feature is positively skewed and value is:", skew_value)
    elif skew_value <= -0.1:
        print("\n", col, "feature is negatively skewed and value is:", skew_value)
    else:
        print("\n", col, "feature is normally distributed and value is:", skew_value)
```

```
 Designation feature is normally distributed and value is: 0.09242138478903683

 Resource Allocation feature is positively skewed and value is: 0.2110787436948646

 Mental Fatigue Score feature is negatively skewed and value is: -0.45245780687704834

 Burn Rate feature is normally distributed and value is: 0.046910742768045674
```

```
# replace the null values with mean
burnoutDf['Resource Allocation'].fillna(burnoutDf['Resource Allocation'].mean(),inplace=True)
burnoutDf['Mental Fatigue Score'].fillna(burnoutDf['Mental Fatigue Score'].mean(),inplace=True)
burnoutDf['Burn Rate'].fillna(burnoutDf['Burn Rate'].mean(),inplace=True)
```

```
#check for null values
burnoutDf.isna().sum()
```

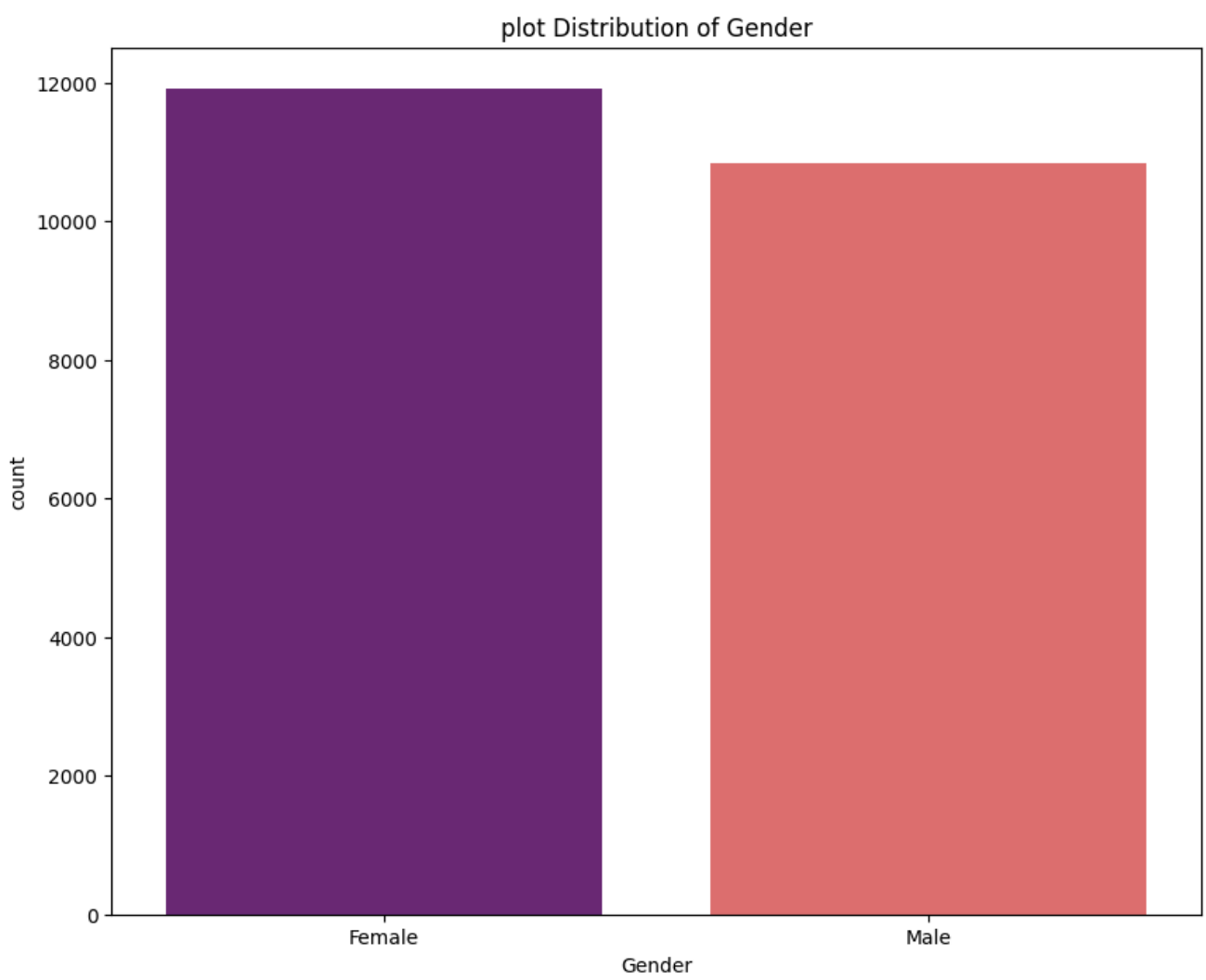
	0
Date of Joining	0
Gender	0
Company Type	0
WFH Setup Available	0
Designation	0
Resource Allocation	0
Mental Fatigue Score	0
Burn Rate	0

```
#burnoutDf.corr()

# Select only numeric columns
numeric_columns = burnoutDf.select_dtypes(include=[int, float])

# Calculate correlation for numeric columns
correlation_matrix = numeric_columns.corr()

# Display the correlation matrix
print(correlation_matrix)
```

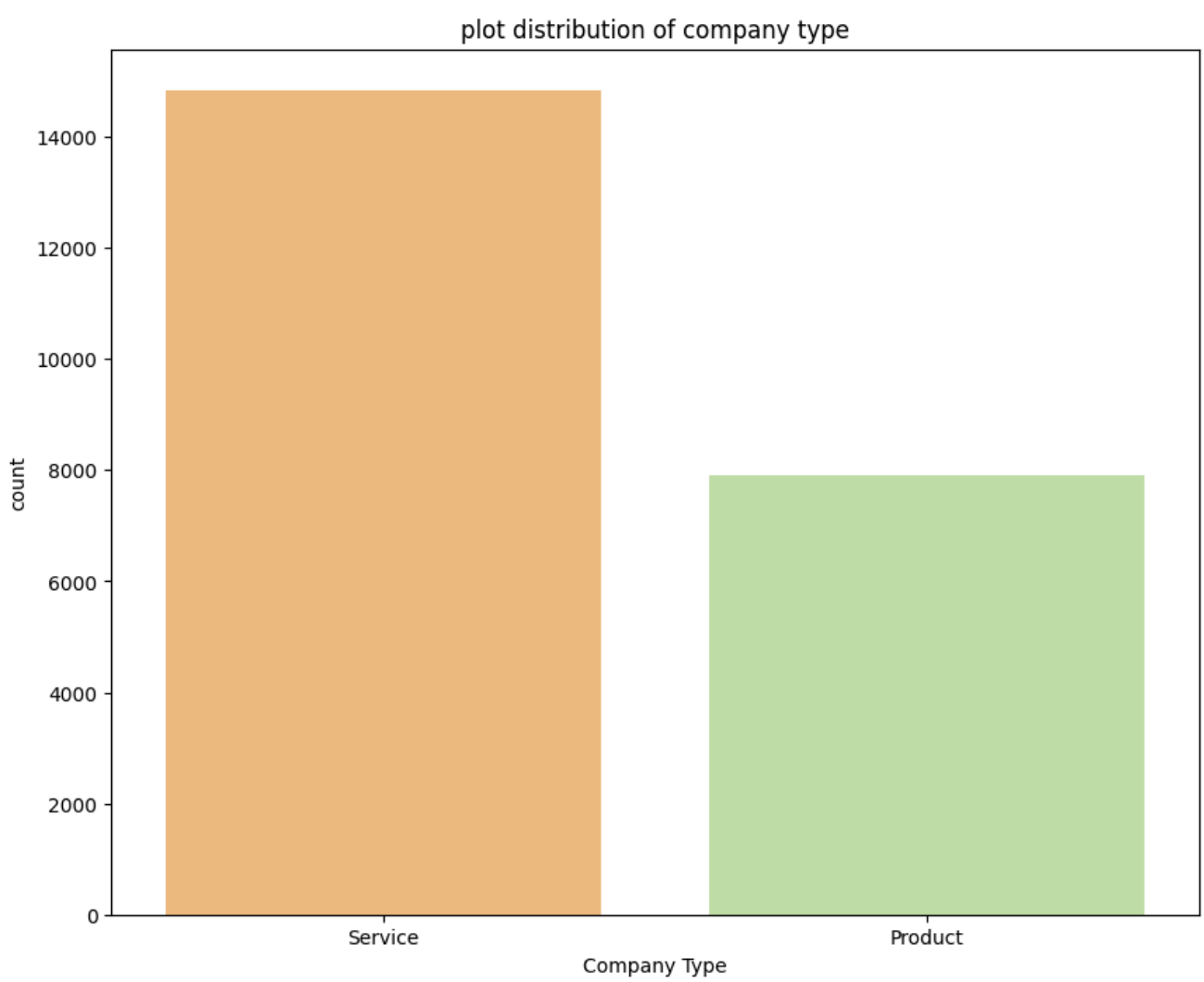
```
                     Designation  Resource Allocation  Mental Fatigue Score  \
Designation             1.000000             0.852046              0.656445
Resource Allocation     0.852046             1.000000              0.739268
Mental Fatigue Score    0.656445             0.739268              1.000000
Burn Rate               0.719284             0.811062              0.878217

                     Burn Rate
Designation           0.719284
Resource Allocation   0.811062
Mental Fatigue Score  0.878217
Burn Rate             1.000000
```

Data Visualization

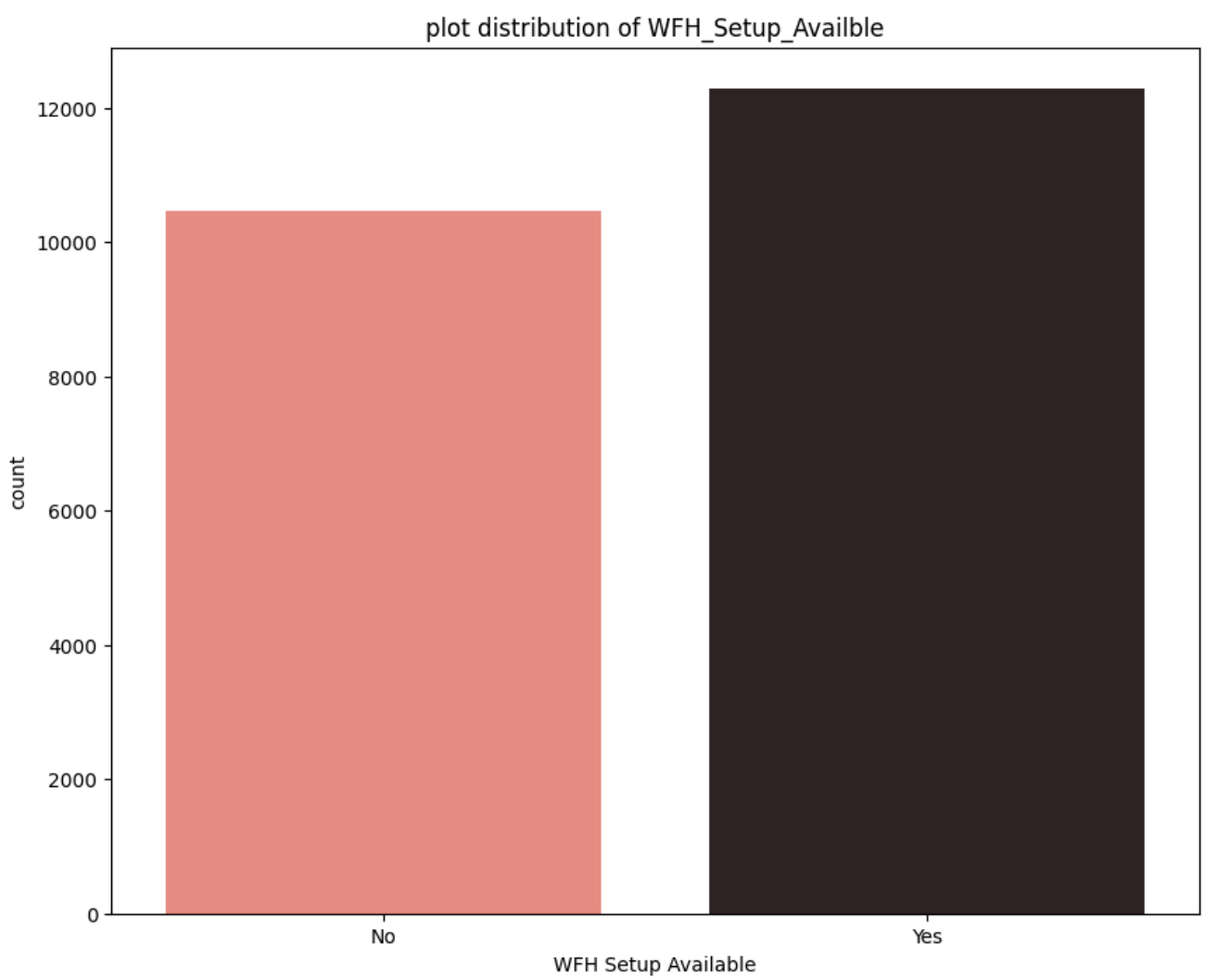
```
#count plot distribution of "Gender"
plt.figure(figsize=(10,8))
sns.countplot(x="Gender",data=burnoutDf,palette="magma")
plt.title("plot Distribution of Gender")
plt.show()
```




```
# count plot distibution of "company type"
plt.figure(figsize=(10,8))
sns.countplot(x="Company Type",data=burnoutDf,palette="Spectral")
plt.title("plot distribution of company type")
plt.show()
```

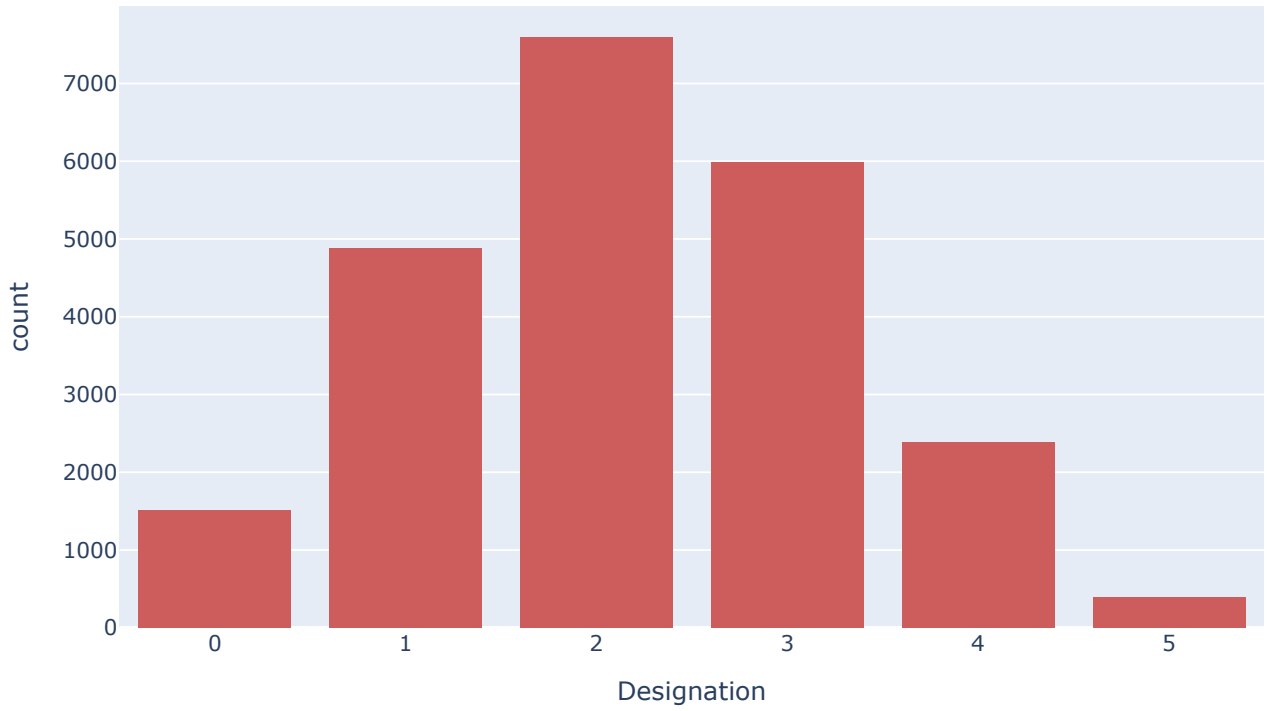



```
#count plot distribution of WFH Setup Availble"
plt.figure(figsize=(10,8))
sns.countplot(x="WFH Setup Available",data=burnoutDf,palette="dark:salmon_r")
plt.title("plot distribution of WFH_Setup_Availble")
plt.show()
```

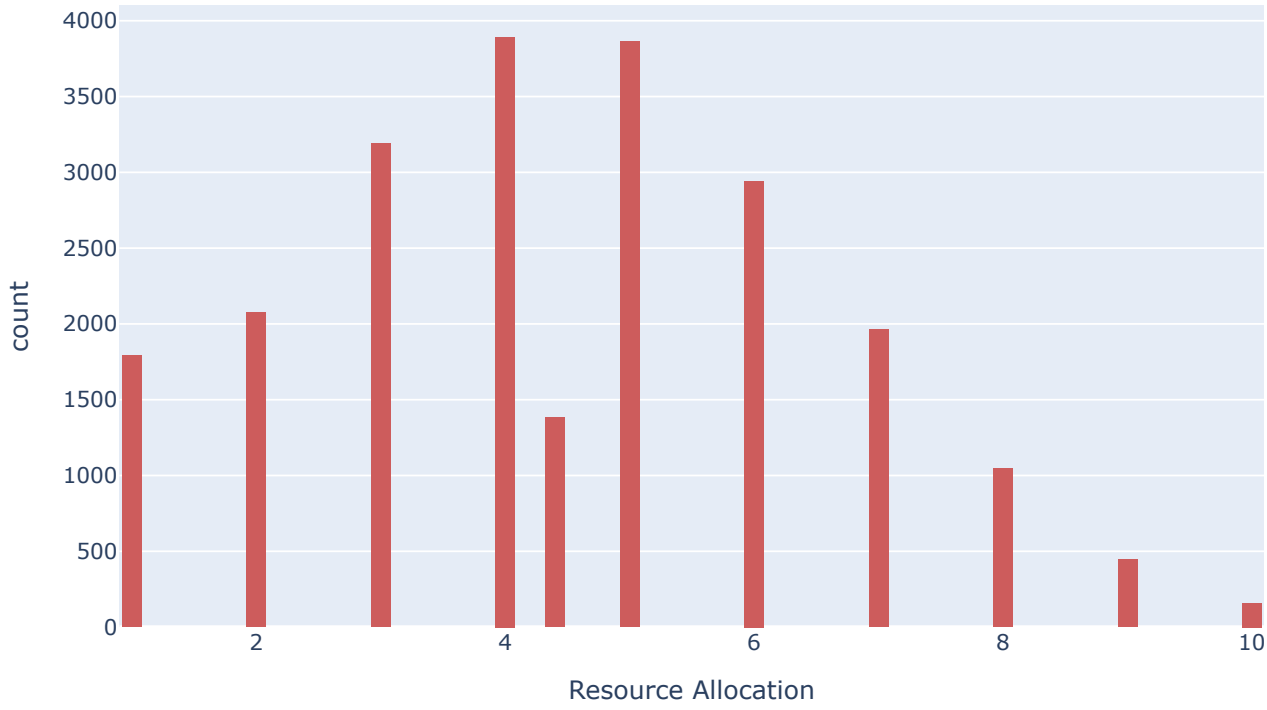



```
# count-Plot Distribution of attributes with the help of Histogram
burn_st=burnoutDf.loc[:,'Date of Joining':'Burn Rate']
burn_st=burn_st.select_dtypes([int,float])
for i, col in enumerate(burn_st.columns):
  fig =px.histogram(burn_st, x=col, title="Plot Distribution of "+col,color_discrete_sequence=["indianred"])
  fig.update_layout(bargap=0.2)
  fig.show()
```

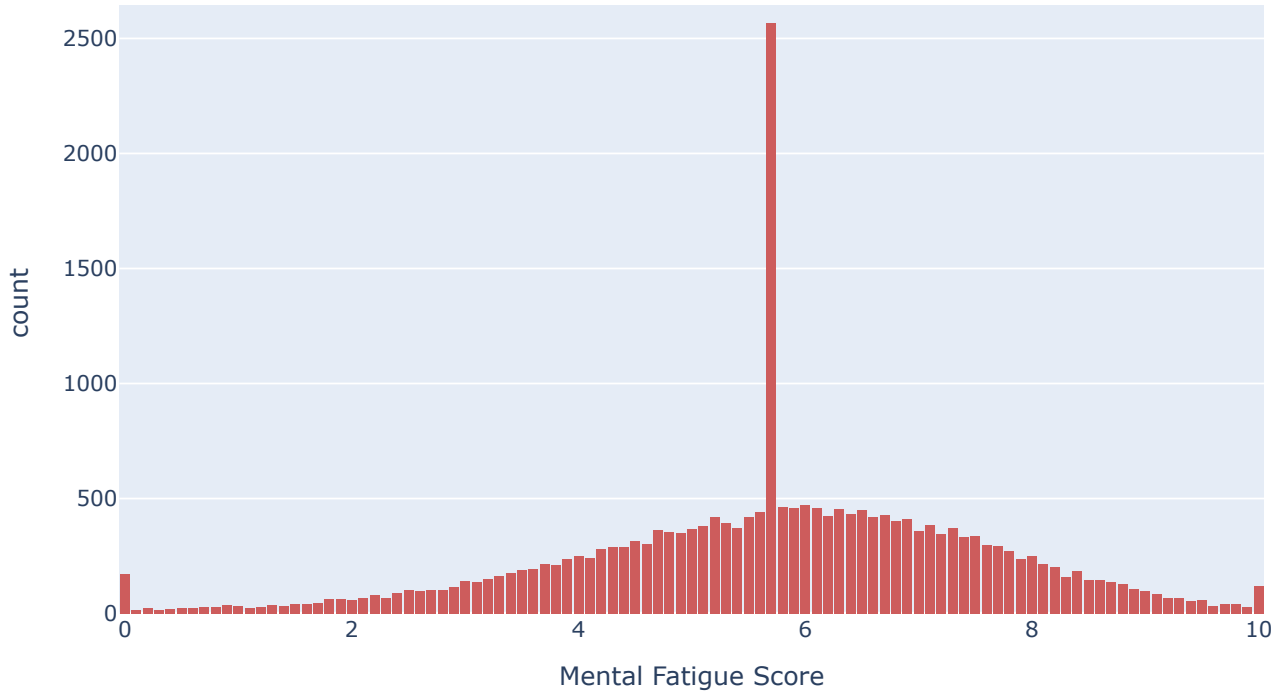
Plot Distribution of Designation



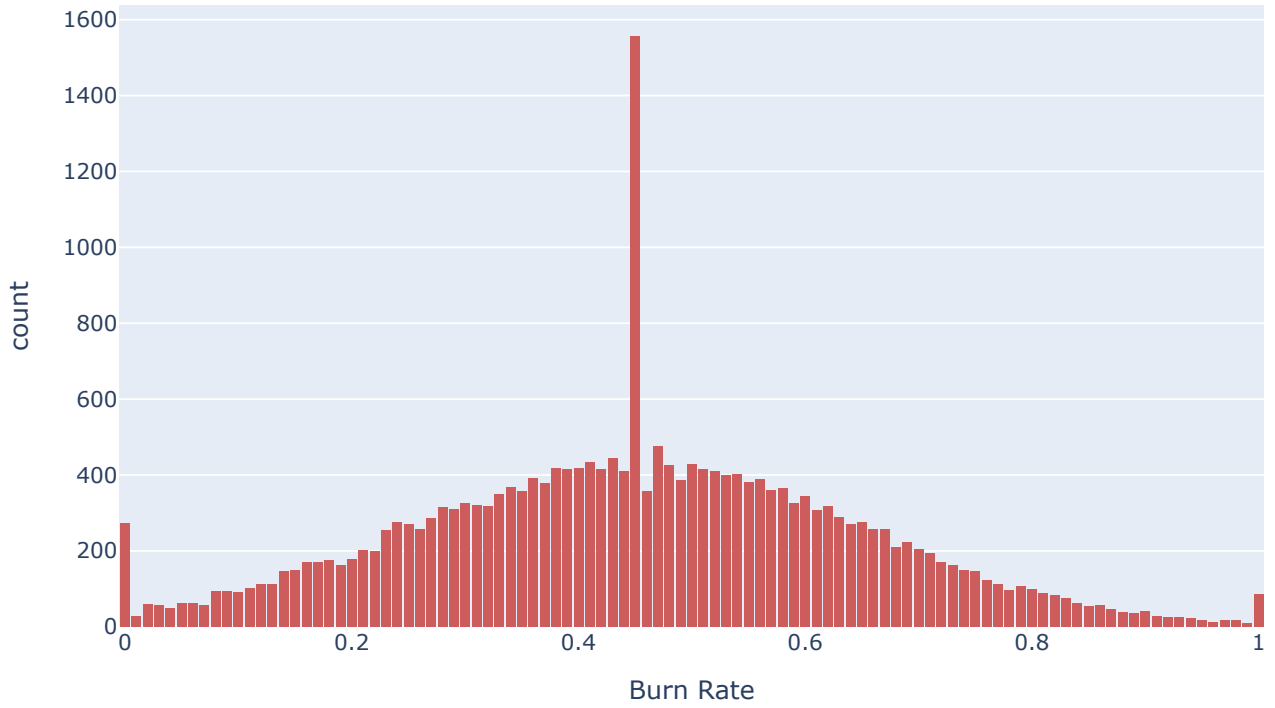
Plot Distribution of Resource Allocation



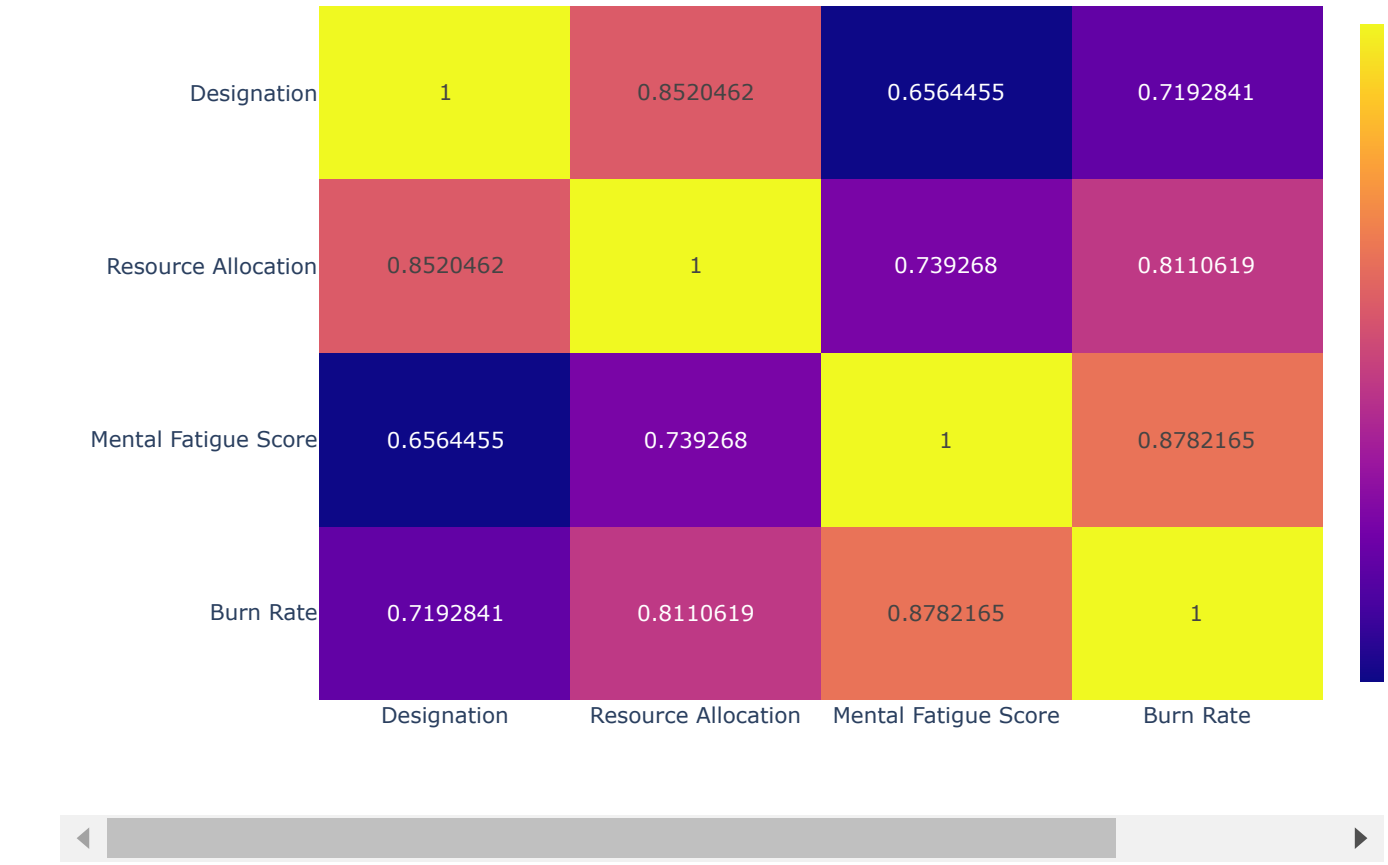
Plot Distribution of Mental Fatigue Score



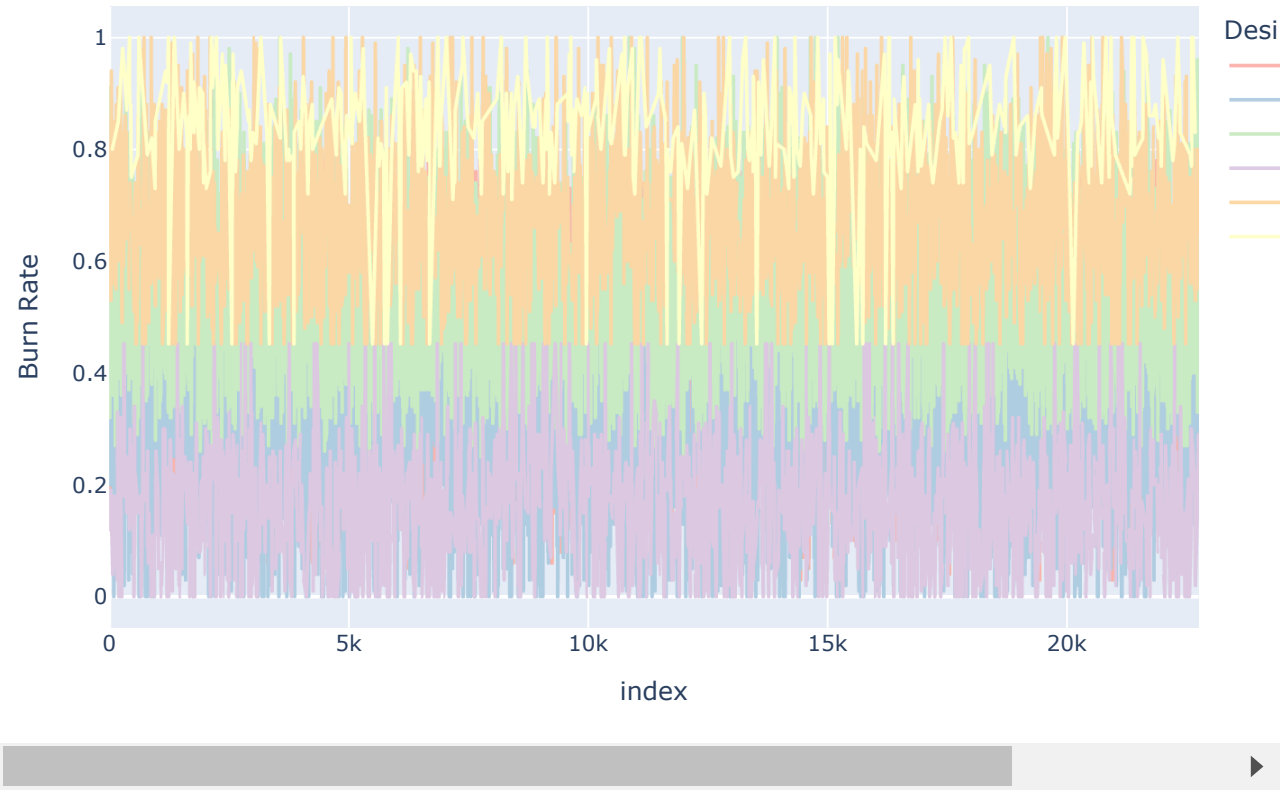
Plot Distribution of Burn Rate



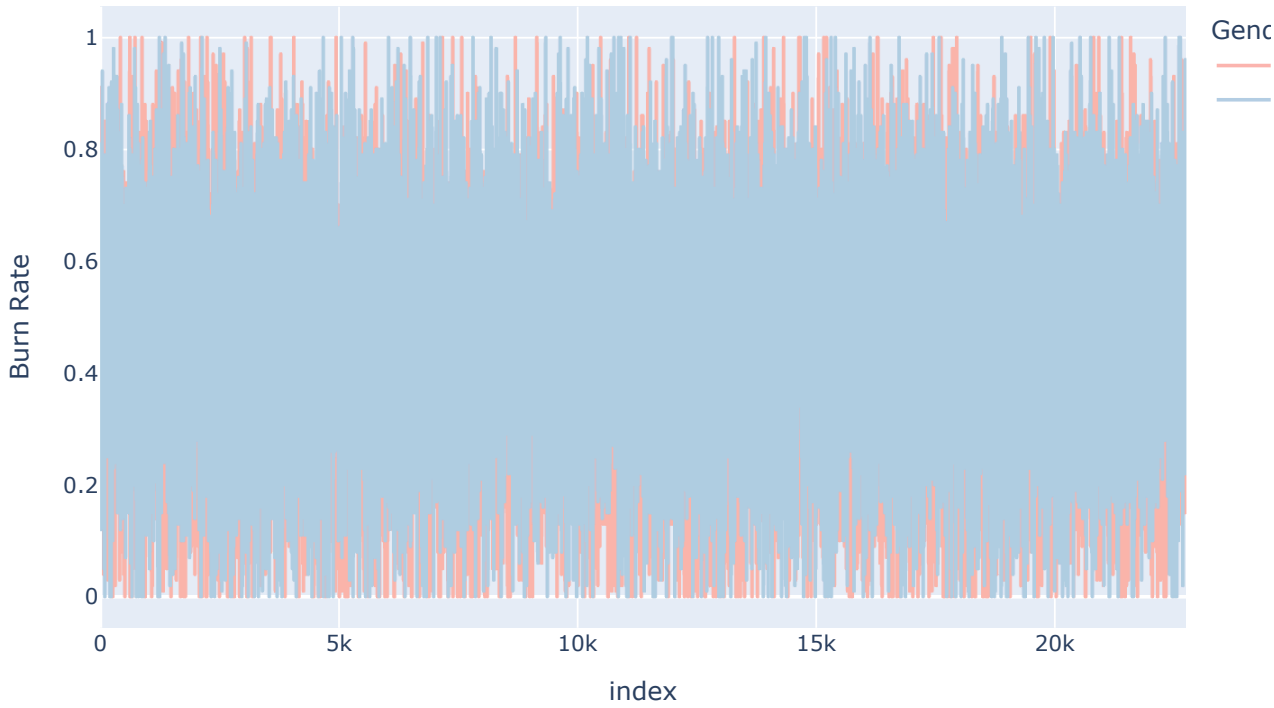
```
import seaborn as sns
import plotly.express as px

# Select only numeric columns
numeric_columns = burnoutDf.select_dtypes(include=[int, float])

# Calculate the correlation matrix for numeric columns
Corr = numeric_columns.corr()

# Plotting heatmap using plotly express
fig = px.imshow(Corr, text_auto=True, aspect='auto')
fig.show()
```



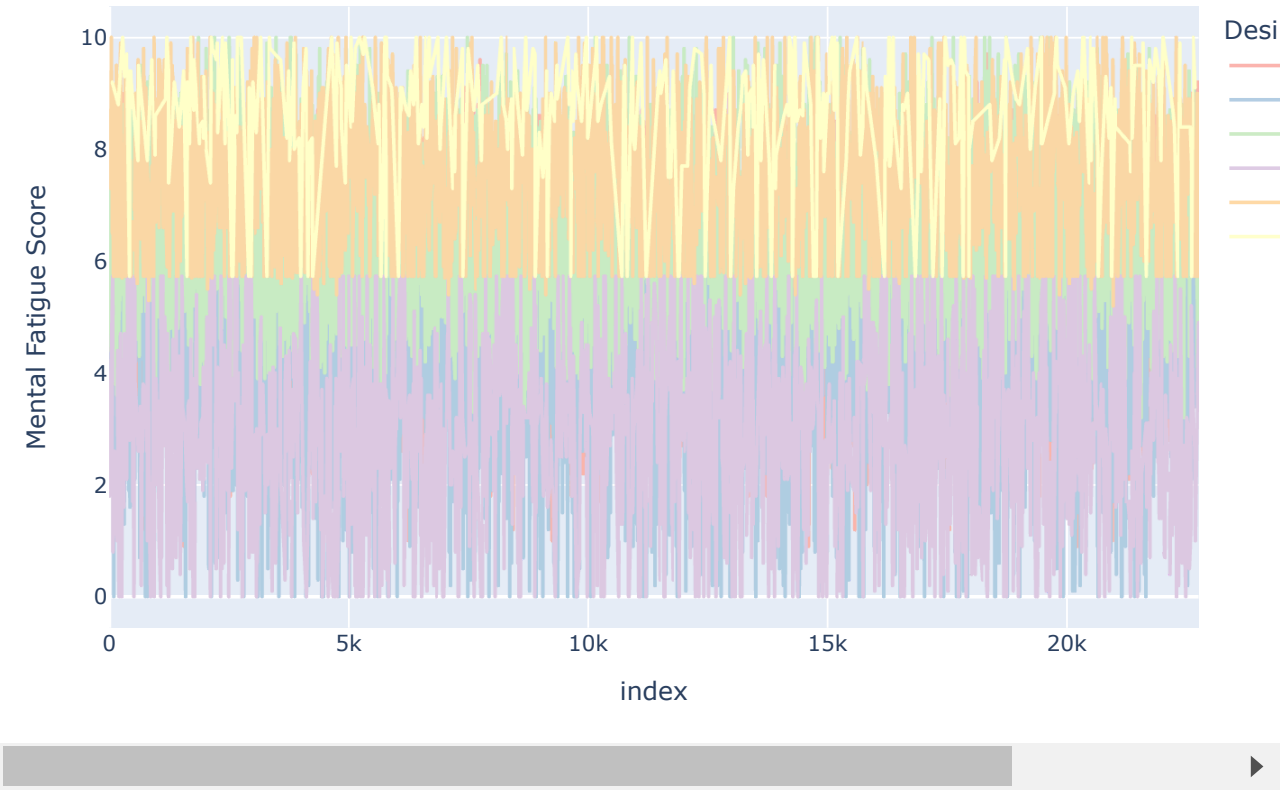

```
ribution of Burn Rate  on the basis of Designation
(burnoutDf, y="Burn Rate", color="Designation",title="Burn rate on the basis of Designation",color_discrete_s
layout(bargap=0.2)
```




```
# plot distribution of Burn Rate  on the basis of Gender
fig=px.line(burnoutDf, y="Burn Rate", color="Gender",title="Burn rate on the basis of Gender",color_discrete_
fig.update_layout(bargap=0.2)
fig.show()
```

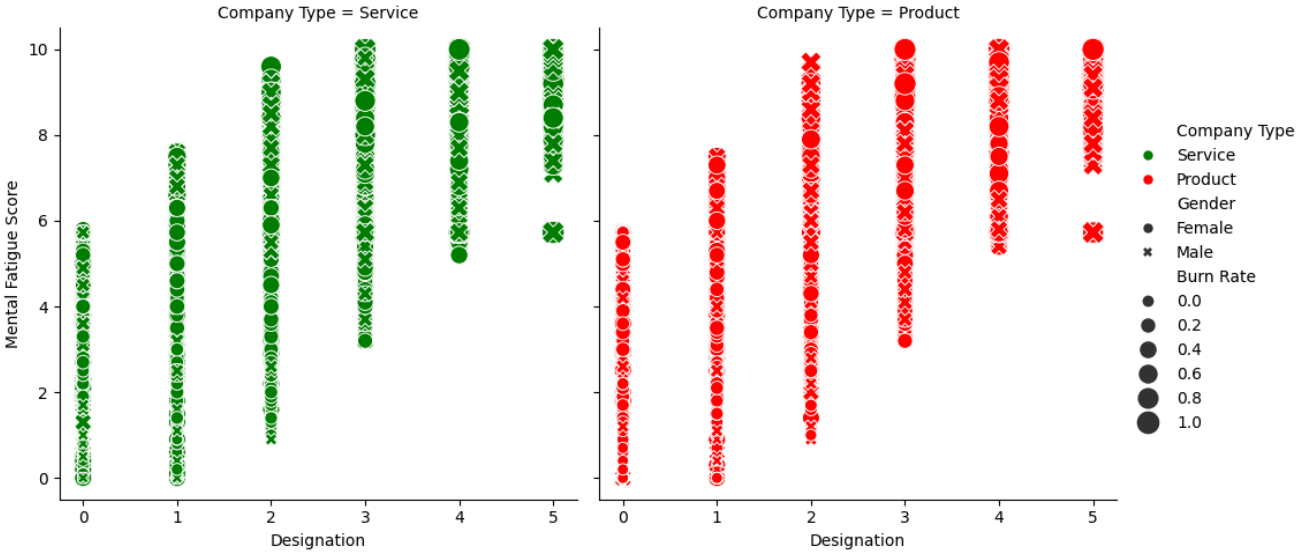



```
# plot distribution of mental fatigue  on the basis of Designation
fig=px.line(burnoutDf, y="Mental Fatigue Score",color="Designation",title="Mental Fatigue vs Designation",co
fig.update_layout(bargap=0.2)
fig.show()
```




```
#Plot Distribution of "Designation" vs mental fatigue" as per Company type ,Burn rate and Gender
sns.relplot(
    data=burnoutDf,x="Designation", y="Mental Fatigue Score",col="Company Type",
    hue="Company Type",size="Burn Rate",style="Gender",
    palette=["g","r"],sizes=(50,200)
)
```

```
<seaborn.axisgrid.FacetGrid at 0x7dd7b06fa6b0>
```



Label Encoding

```
# label encoding and assign in new variable
from sklearn.preprocessing
```

```
from sklearn import preprocessing
Label_encode = preprocessing.LabelEncoder()


# Assign in new variable
burnoutDf['GenderLabel'] = Label_encode.fit_transform(burnoutDf['Gender'].values)
burnoutDf['Company_TypeLabel'] = Label_encode.fit_transform(burnoutDf['Company Type'].values)
burnoutDf['WFH_Setup_AvailableLabel'] = Label_encode.fit_transform(burnoutDf['WFH Setup Available'].values)


#check assigned values
gn=burnoutDf.groupby('Gender')
gn=gn['GenderLabel']
gn.first()
```

Gender	GenderLabel
Female	0
Male	1

dtype: int64

```
#check assigned values
ct=burnoutDf.groupby('Company Type')
ct=ct['Company_TypeLabel']
ct.first()
```

Company Type	Company_TypeLabel
Product	0
Service	1

```
# check assigned values
wsa=burnoutDf.groupby('WFH Setup Available')
wsa=wsa['WFH_Setup_AvailableLabel']
wsa.first()
```

WFH Setup Available	WFH_Setup_AvailableLabel
No	0
Yes	1