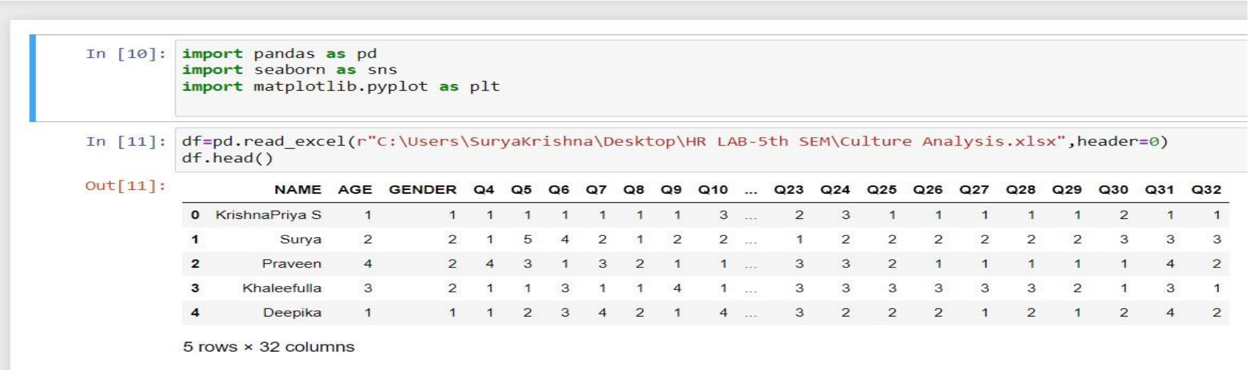
ANNEXTURE

Contents:

- Culture Analysis.
- Talent Management Analysis.
- Competency Analysis.
- Cost and Productivity Analysis.
- Recruitment Analysis.
- Performance Management Analysis.
- Training and Development Analysis.

Code & Output:

Culture Analysis:



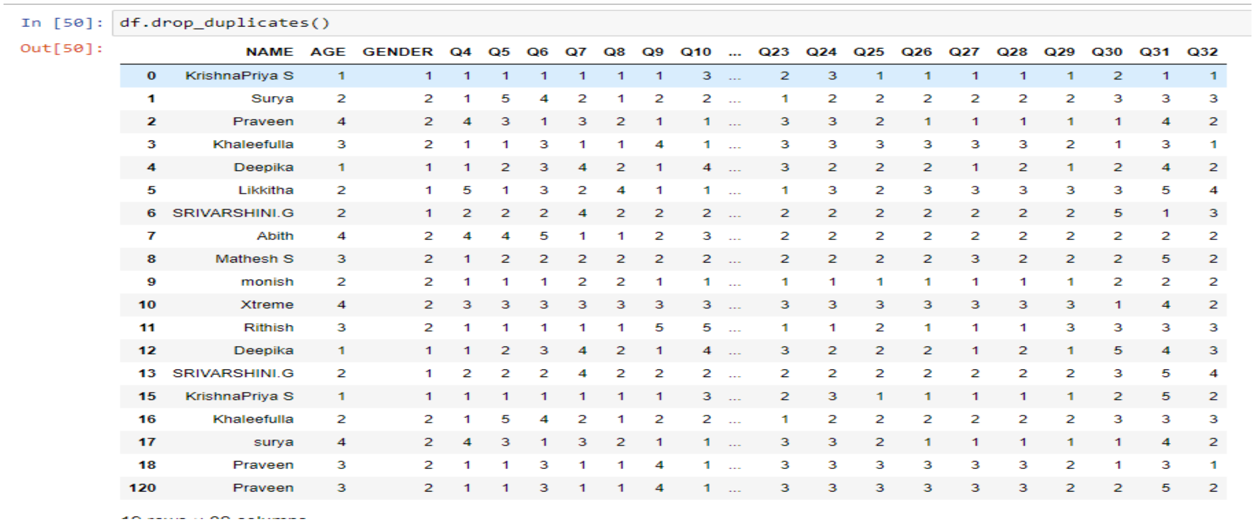
In [10]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [11]:
```
df=pd.read_excel(r"C:\Users\SuryaKrishna\Desktop\HR LAB-5th SEM\Culture Analysis.xlsx",header=0)
df.head()
```

Out[11]:

	NAME	AGE	GENDER	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32
0	KrishnaPriya S	1	1	1	1	1	1	1	1	3	...	2	3	1	1	1	1	1	2	1	1
1	Surya	2	2	1	5	4	2	1	2	2	...	1	2	2	2	2	2	2	3	3	3
2	Praveen	4	2	4	3	1	3	2	1	1	...	3	3	2	1	1	1	1	1	4	2
3	Khaleefulla	3	2	1	1	3	1	1	4	1	...	3	3	3	3	3	3	2	1	3	1
4	Deepika	1	1	1	2	3	4	2	1	4	...	3	2	2	2	1	2	1	2	4	2

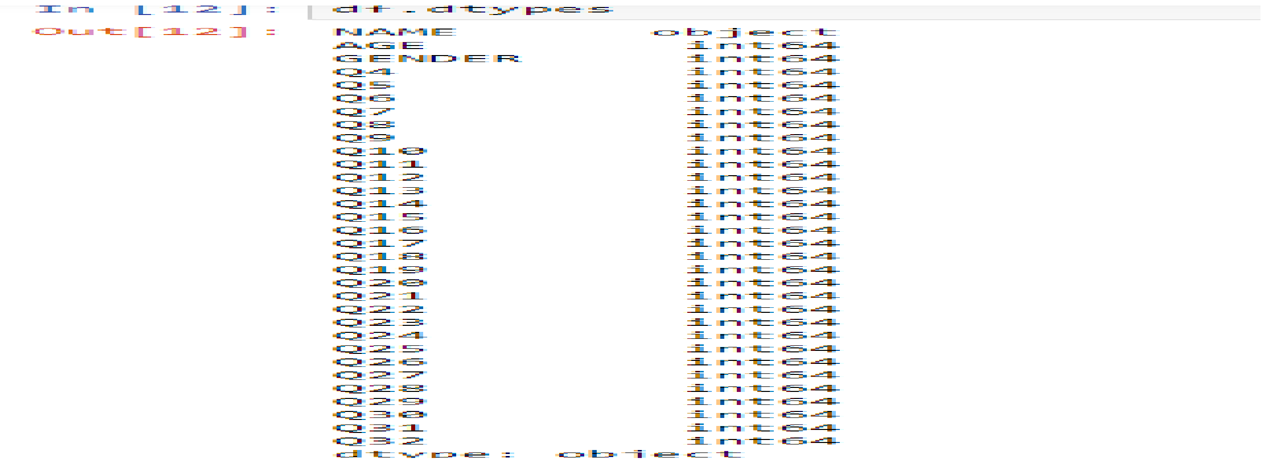
5 rows × 32 columns



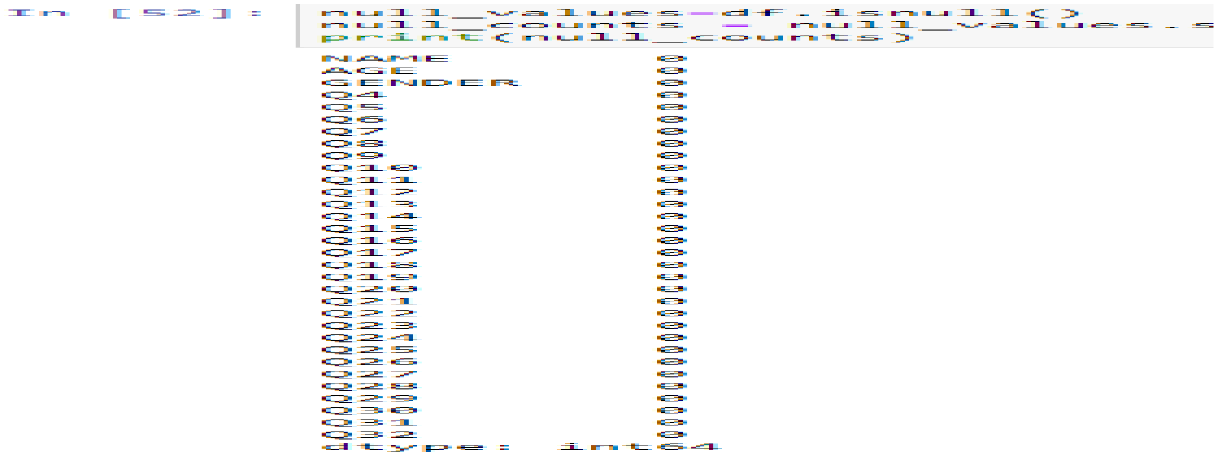
In [50]:
```
df.drop_duplicates()
```

Out[50]:

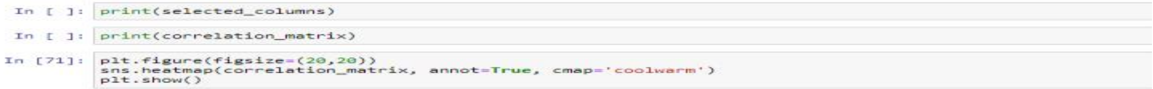
	NAME	AGE	GENDER	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32
0	KrishnaPriya S	1	1	1	1	1	1	1	1	3	...	2	3	1	1	1	1	1	2	1	1
1	Surya	2	2	1	5	4	2	1	2	2	...	1	2	2	2	2	2	2	3	3	3
2	Praveen	4	2	4	3	1	3	2	1	1	...	3	3	2	1	1	1	1	1	4	2
3	Khaleefulla	3	2	1	1	3	1	1	4	1	...	3	3	3	3	3	3	2	1	3	1
4	Deepika	1	1	1	2	3	4	2	1	4	...	3	2	2	2	1	2	1	2	4	2
5	Likkitha	2	1	5	1	3	2	4	1	1	...	1	3	2	3	3	3	3	3	5	4
6	SRIVARSHINI.G	2	1	2	2	2	4	2	2	2	...	2	2	2	2	2	2	2	5	1	3
7	Abith	4	2	4	4	5	1	1	2	3	...	2	2	2	2	2	2	2	2	2	2
8	Mathesh S	3	2	1	2	2	2	2	2	2	...	2	2	2	2	3	2	2	2	5	2
9	monish	2	2	1	1	1	2	2	1	1	...	1	1	1	1	1	1	1	2	2	2
10	Xtreme	4	2	3	3	3	3	3	3	3	...	3	3	3	3	3	3	3	1	4	2
11	Rithish	3	2	1	1	1	1	1	5	5	...	1	1	2	1	1	1	3	3	3	3
12	Deepika	1	1	1	2	3	4	2	1	4	...	3	2	2	2	1	2	1	5	4	3
13	SRIVARSHINI.G	2	1	2	2	2	4	2	2	2	...	2	2	2	2	2	2	2	3	5	4
15	KrishnaPriya S	1	1	1	1	1	1	1	1	3	...	2	3	1	1	1	1	1	2	5	2
16	Khaleefulla	2	2	1	5	4	2	1	2	2	...	1	2	2	2	2	2	2	3	3	3
17	surya	4	2	4	3	1	3	2	1	1	...	3	3	2	1	1	1	1	1	4	2
18	Praveen	3	2	1	1	3	1	1	4	1	...	3	3	3	3	3	3	2	1	3	1
120	Praveen	3	2	1	1	3	1	1	4	1	...	3	3	3	3	3	3	2	2	5	2



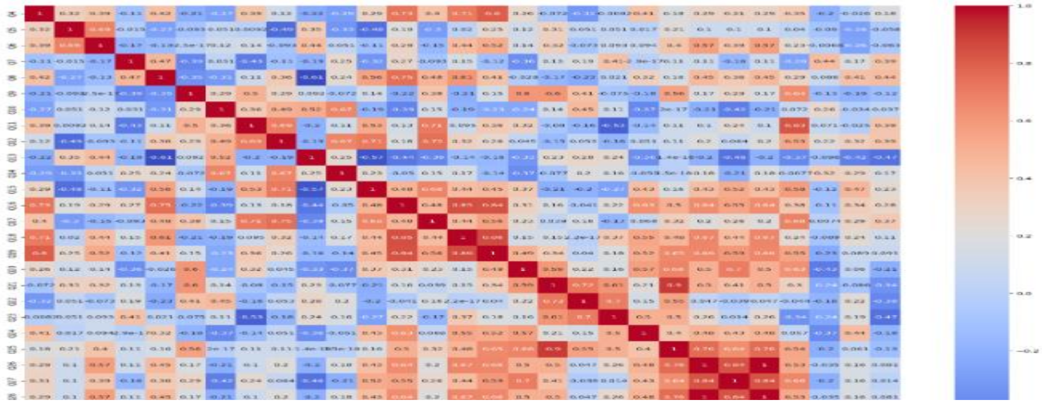
```
In [12]:  df.dtypes
Out[12]:  NAME         object
          AGE           int64
          GENDER        int64
          Q4            int64
          Q5            int64
          Q6            int64
          Q7            int64
          Q8            int64
          Q9            int64
          Q10           int64
          Q11           int64
          Q12           int64
          Q13           int64
          Q14           int64
          Q15           int64
          Q16           int64
          Q17           int64
          Q18           int64
          Q19           int64
          Q20           int64
          Q21           int64
          Q22           int64
          Q23           int64
          Q24           int64
          Q25           int64
          Q26           int64
          Q27           int64
          Q28           int64
          Q29           int64
          Q30           int64
          Q31           int64
          Q32           int64
          dtype: object
```

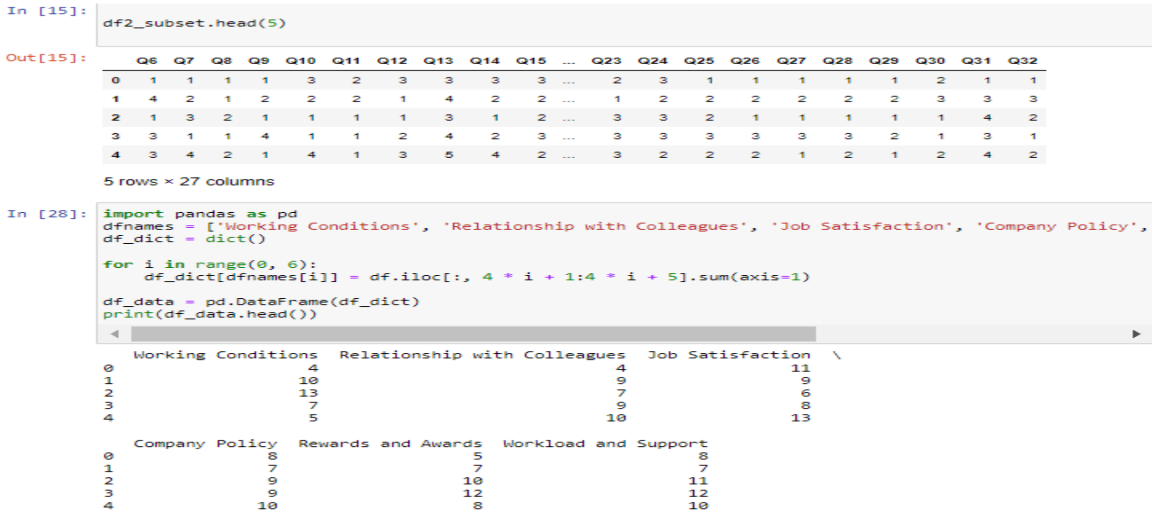


```
In  [52]:  null_values=df.isnull()
           null_counts = null_values.s
           print(null_counts)

            NAME        0
            AGE         0
            GENDER      0
            Q4          0
            Q5          0
            Q6          0
            Q7          0
            Q8          0
            Q9          0
            Q10         0
            Q11         0
            Q12         0
            Q13         0
            Q14         0
            Q15         0
            Q16         0
            Q17         0
            Q18         0
            Q19         0
            Q20         0
            Q21         0
            Q22         0
            Q23         0
            Q24         0
            Q25         0
            Q26         0
            Q27         0
            Q28         0
            Q29         0
            Q30         0
            Q31         0
            Q32         0
            dtype: int64
```



```
In [ ]:  print(selected_columns)

In [ ]:  print(correlation_matrix)

In [71]: plt.figure(figsize=(20,20))
         sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
         plt.show()
```





In [15]:
```
df2_subset.head(5)
```

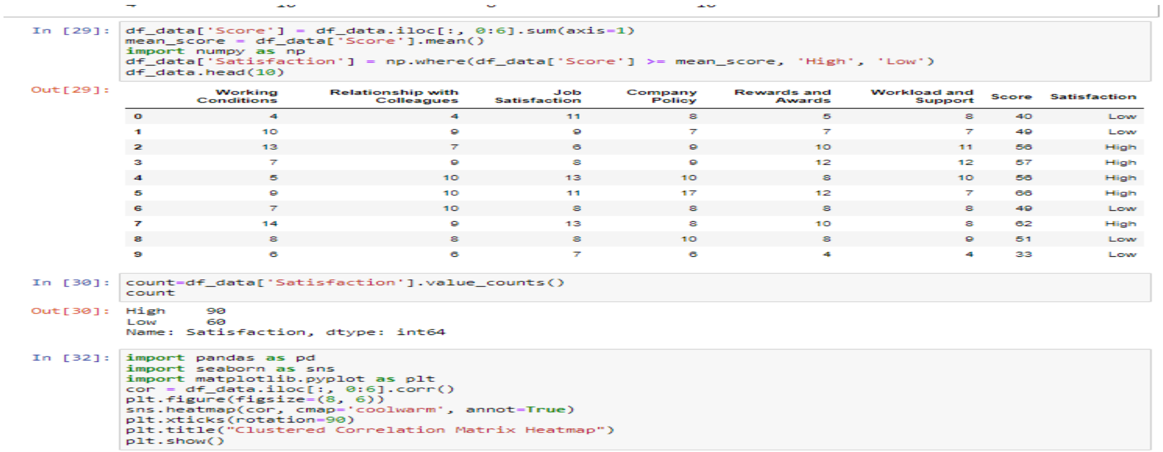
Out[15]:

	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	...	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32
0	1	1	1	1	3	2	3	3	3	3	...	2	3	1	1	1	1	1	2	1	1
1	4	2	1	2	2	2	1	4	2	2	...	1	2	2	2	2	2	2	3	3	3
2	1	3	2	1	1	1	1	3	1	2	...	3	3	2	1	1	1	1	1	4	2
3	3	1	1	4	1	1	2	4	2	3	...	3	3	3	3	3	3	2	1	3	1
4	3	4	2	1	4	1	3	5	4	2	...	3	2	2	2	1	2	1	2	4	2

5 rows × 27 columns

In [28]:
```
import pandas as pd
dfnames = ['Working Conditions', 'Relationship with Colleagues', 'Job Satisfaction', 'Company Policy',
df_dict = dict()

for i in range(0, 6):
    df_dict[dfnames[i]] = df.iloc[:, 4 * i + 1:4 * i + 5].sum(axis=1)

df_data = pd.DataFrame(df_dict)
print(df_data.head())
```

```
   Working Conditions  Relationship with Colleagues  Job Satisfaction  \
0                   4                             4                11   
1                  10                             9                 9   
2                  13                             7                 6   
3                   7                             9                 8   
4                   5                            10                13   

   Company Policy  Rewards and Awards  Workload and Support  
0               8                   5                     8  
1               7                   7                     7  
2               9                  10                    11  
3               9                  12                    12  
4              10                   8                    10  
```



In [29]:
```
df_data['Score'] = df_data.iloc[:, 0:6].sum(axis=1)
mean_score = df_data['Score'].mean()
import numpy as np
df_data['Satisfaction'] = np.where(df_data['Score'] >= mean_score, 'High', 'Low')
df_data.head(10)
```

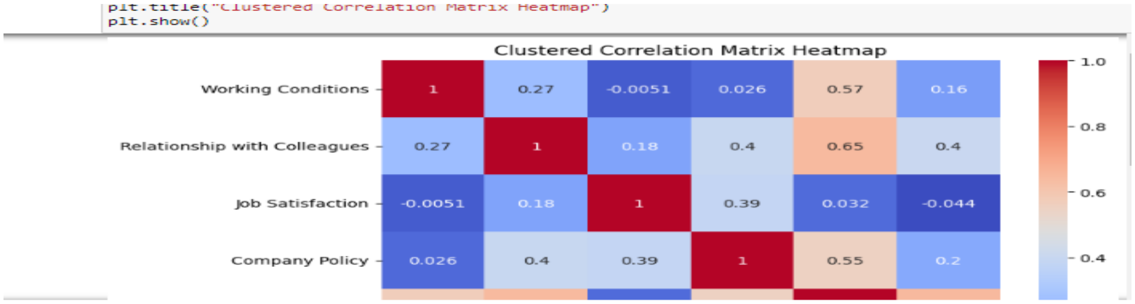
Out[29]:

	Working Conditions	Relationship with Colleagues	Job Satisfaction	Company Policy	Rewards and Awards	Workload and Support	Score	Satisfaction
0	4	4	11	8	5	8	40	Low
1	10	9	9	7	7	7	49	Low
2	13	7	6	9	10	11	56	High
3	7	9	8	9	12	12	57	High
4	5	10	13	10	8	10	56	High
5	9	10	11	17	12	7	66	High
6	7	10	8	8	8	8	49	Low
7	14	9	13	8	10	8	62	High
8	8	8	8	10	8	9	51	Low
9	6	6	7	6	4	4	33	Low

In [30]:
```
count=df_data['Satisfaction'].value_counts()
count
```

Out[30]:
```
High     90
Low      60
Name: Satisfaction, dtype: int64
```

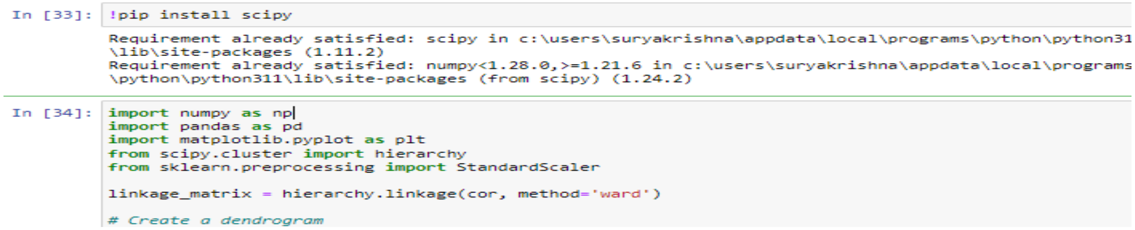
In [32]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
cor = df_data.iloc[:, 0:6].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(cor, cmap='coolwarm', annot=True)
plt.xticks(rotation=90)
plt.title("Clustered Correlation Matrix Heatmap")
plt.show()
```



```
plt.title("Clustered Correlation Matrix Heatmap")
plt.show()
```

Clustered Correlation Matrix Heatmap

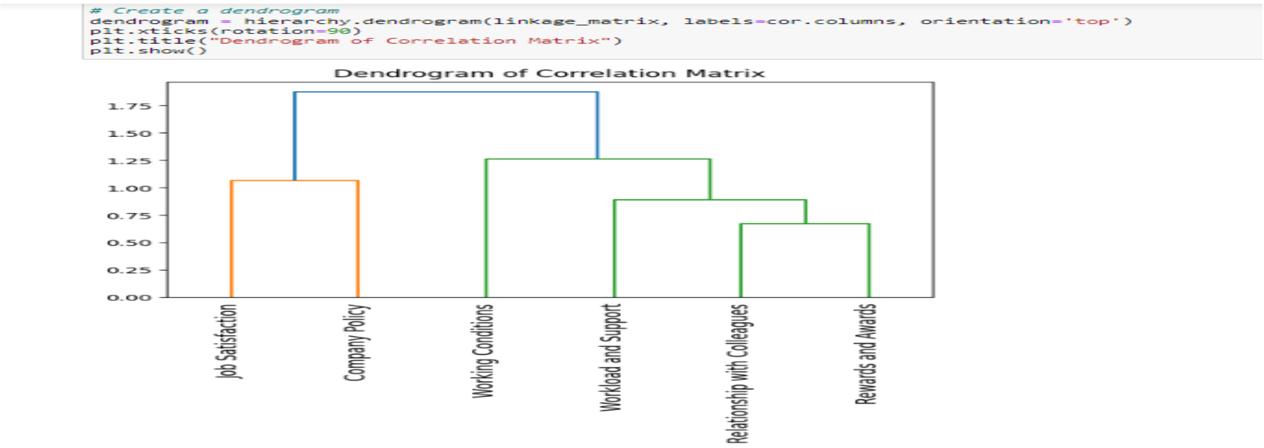
Working Conditions: 1, 0.27, -0.0051, 0.026, 0.57, 0.16
Relationship with Colleagues: 0.27, 1, 0.18, 0.4, 0.65, 0.4
Job Satisfaction: -0.0051, 0.18, 1, 0.39, 0.032, -0.044
Company Policy: 0.026, 0.4, 0.39, 1, 0.55, 0.2
1.0
0.8
0.6
0.4



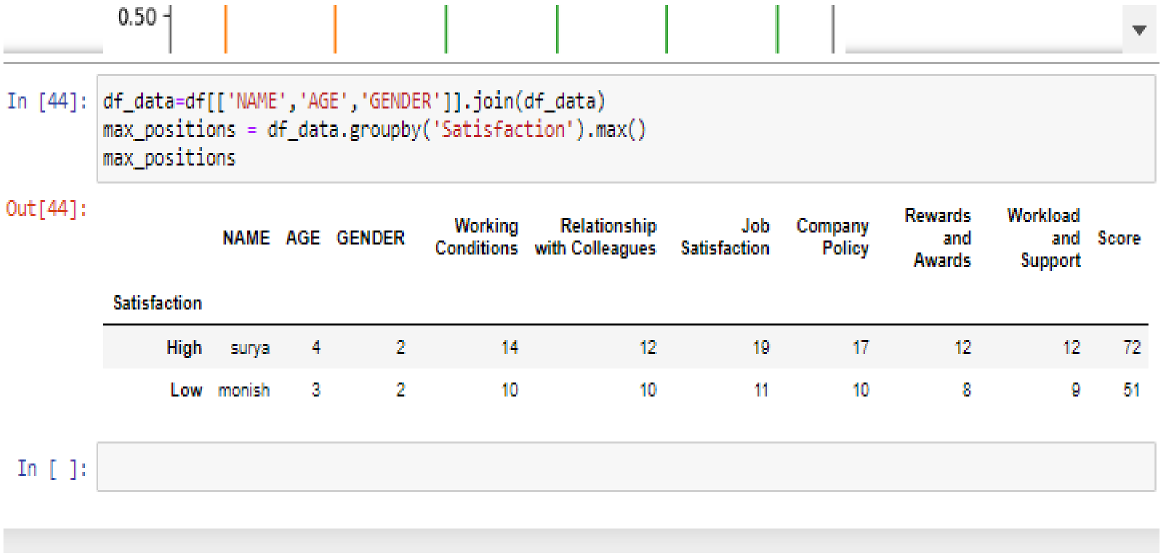
In [33]:
```
!pip install scipy
```

```
Requirement already satisfied: scipy in c:\users\suryakrishna\appdata\local\programs\python\python31
\lib\site-packages (1.11.2)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in c:\users\suryakrishna\appdata\local\programs
\python\python311\lib\site-packages (from scipy) (1.24.2)
```

In [34]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster import hierarchy
from sklearn.preprocessing import StandardScaler

linkage_matrix = hierarchy.linkage(cor, method='ward')

# Create a dendrogram
```



```
# Create a dendrogram
dendrogram = hierarchy.dendrogram(linkage_matrix, labels=cor.columns, orientation='top')
plt.xticks(rotation=90)
plt.title("Dendrogram of Correlation Matrix")
plt.show()
```



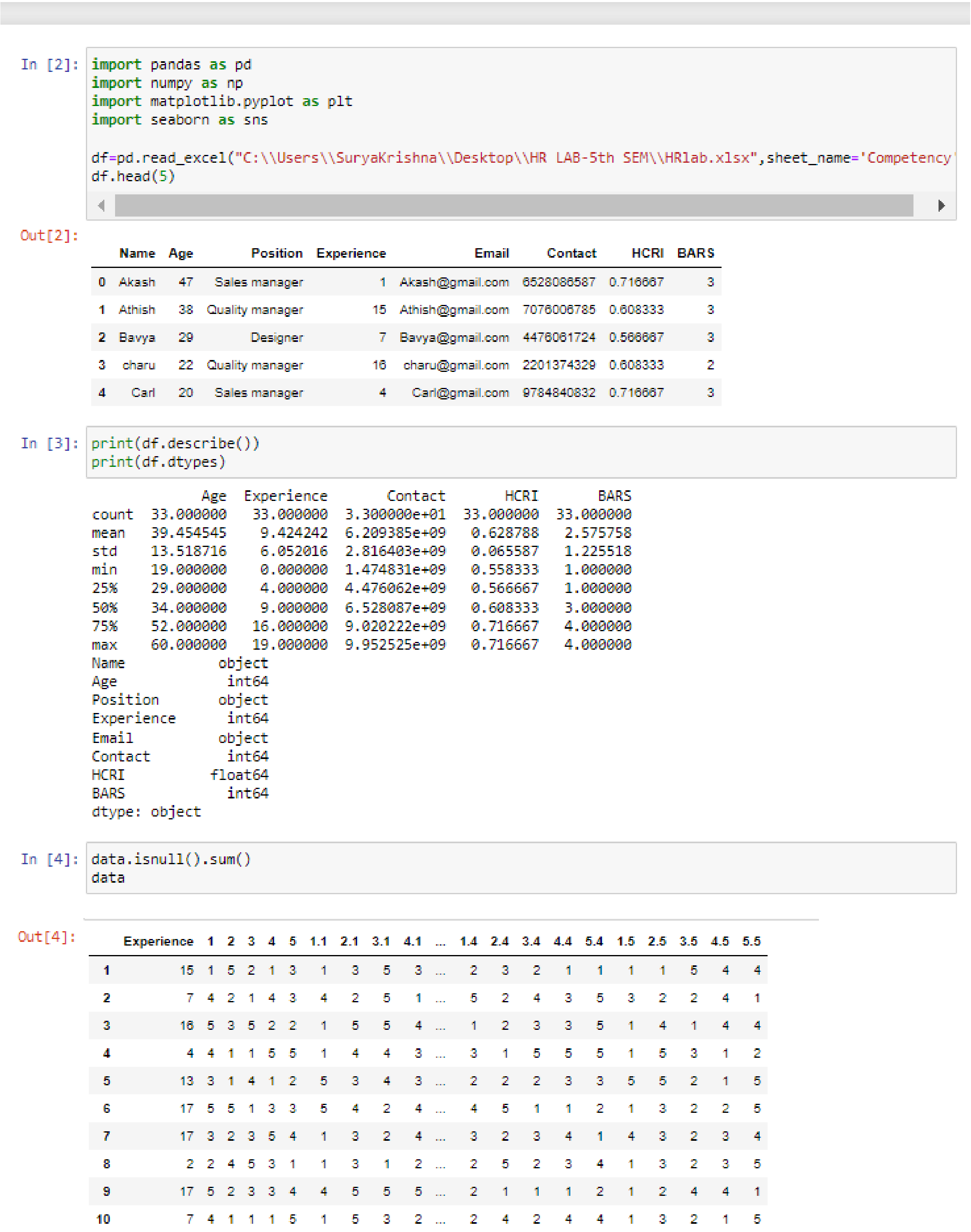
```
In [44]: df_data=df[['NAME','AGE','GENDER']].join(df_data)
         max_positions = df_data.groupby('Satisfaction').max()
         max_positions
```

Out[44]:

	NAME	AGE	GENDER	Working Conditions	Relationship with Colleagues	Job Satisfaction	Company Policy	Rewards and Awards	Workload and Support	Score
Satisfaction										
High	surya	4	2	14	12	19	17	12	12	72
Low	monish	3	2	10	10	11	10	8	9	51

```
In [ ]:
```

Competency Analysis:



In [2]:

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

df=pd.read_excel("C:\\Users\\SuryaKrishna\\Desktop\\HR LAB-5th SEM\\HRlab.xlsx",sheet_name='Competency'
df.head(5)
```

Out[2]:

	Name	Age	Position	Experience	Email	Contact	HCRI	BARS
0	Akash	47	Sales manager	1	Akash@gmail.com	6528086587	0.716667	3
1	Athish	38	Quality manager	15	Athish@gmail.com	7076006785	0.608333	3
2	Bavya	29	Designer	7	Bavya@gmail.com	4476061724	0.566667	3
3	charu	22	Quality manager	16	charu@gmail.com	2201374329	0.608333	2
4	Carl	20	Sales manager	4	Carl@gmail.com	9784840832	0.716667	3

In [3]:

```
print(df.describe())
print(df.dtypes)
```

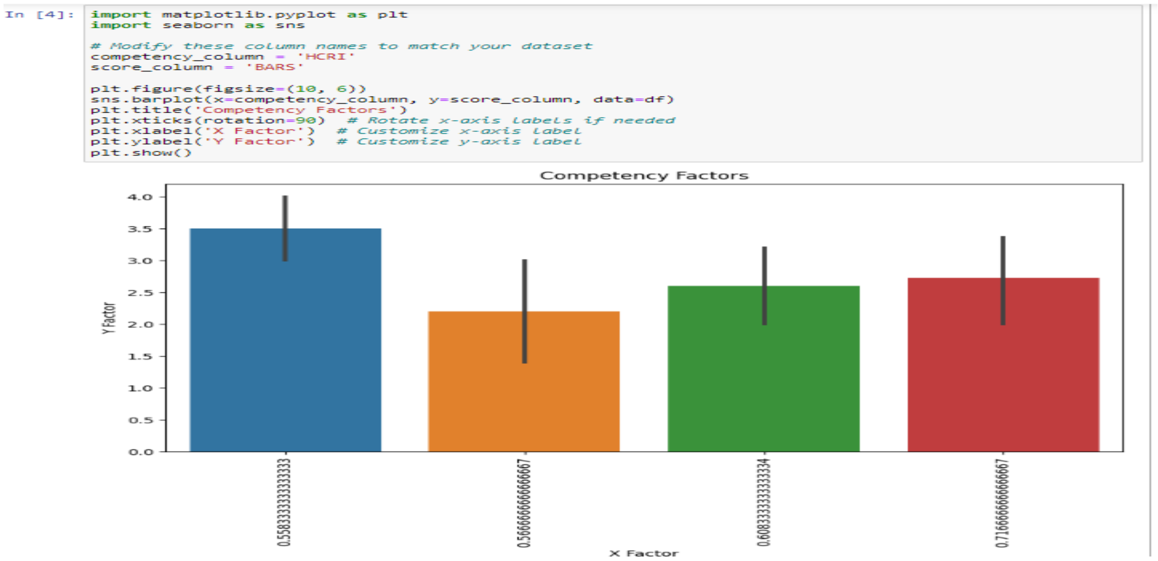
```
             Age  Experience       Contact       HCRI       BARS
count  33.000000   33.000000  3.300000e+01  33.000000  33.000000
mean   39.454545    9.424242  6.209385e+09   0.628788   2.575758
std    13.518716    6.052016  2.816403e+09   0.065587   1.225518
min    19.000000    0.000000  1.474831e+09   0.558333   1.000000
25%    29.000000    4.000000  4.476062e+09   0.566667   1.000000
50%    34.000000    9.000000  6.528087e+09   0.608333   3.000000
75%    52.000000   16.000000  9.020222e+09   0.716667   4.000000
max    60.000000   19.000000  9.952525e+09   0.716667   4.000000
Name           object
Age             int64
Position       object
Experience      int64
Email          object
Contact         int64
HCRI          float64
BARS            int64
dtype: object
```

In [4]:

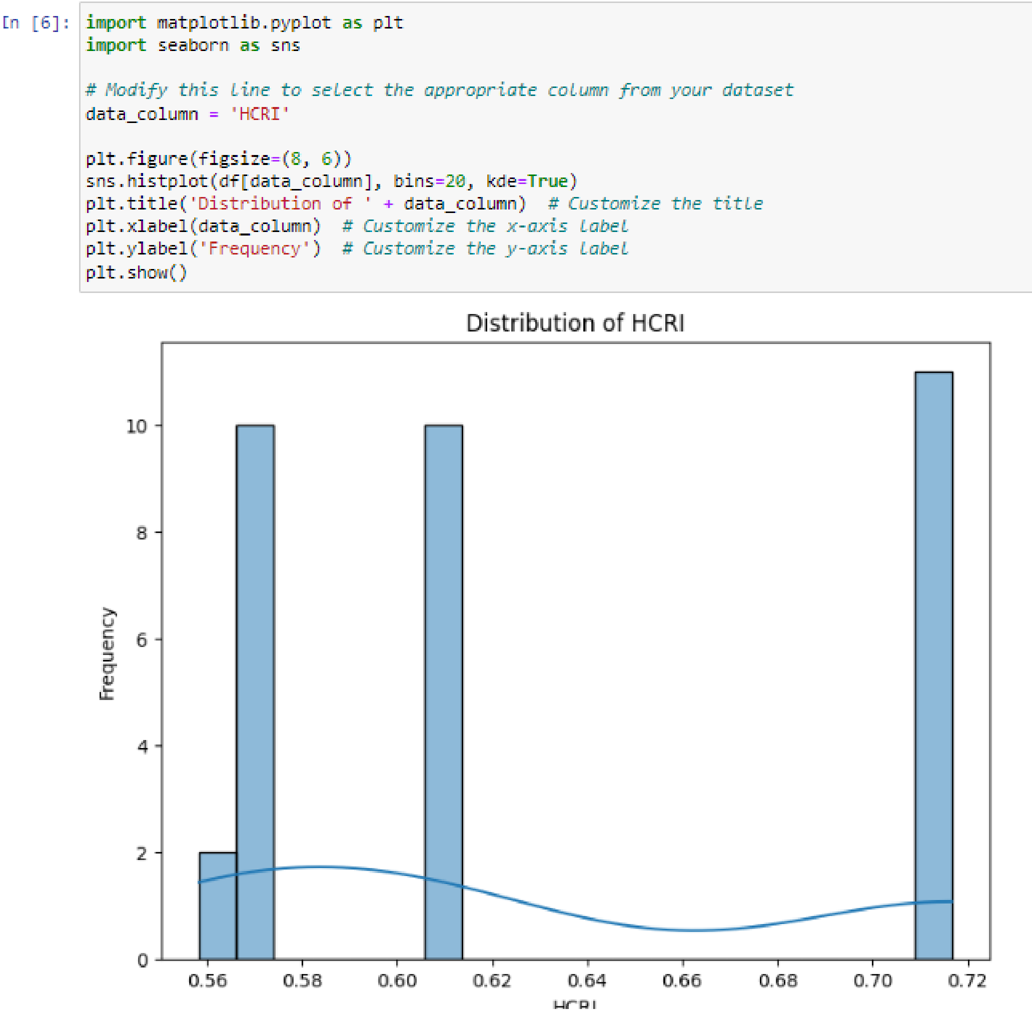
```
data.isnull().sum()
data
```

Out[4]:

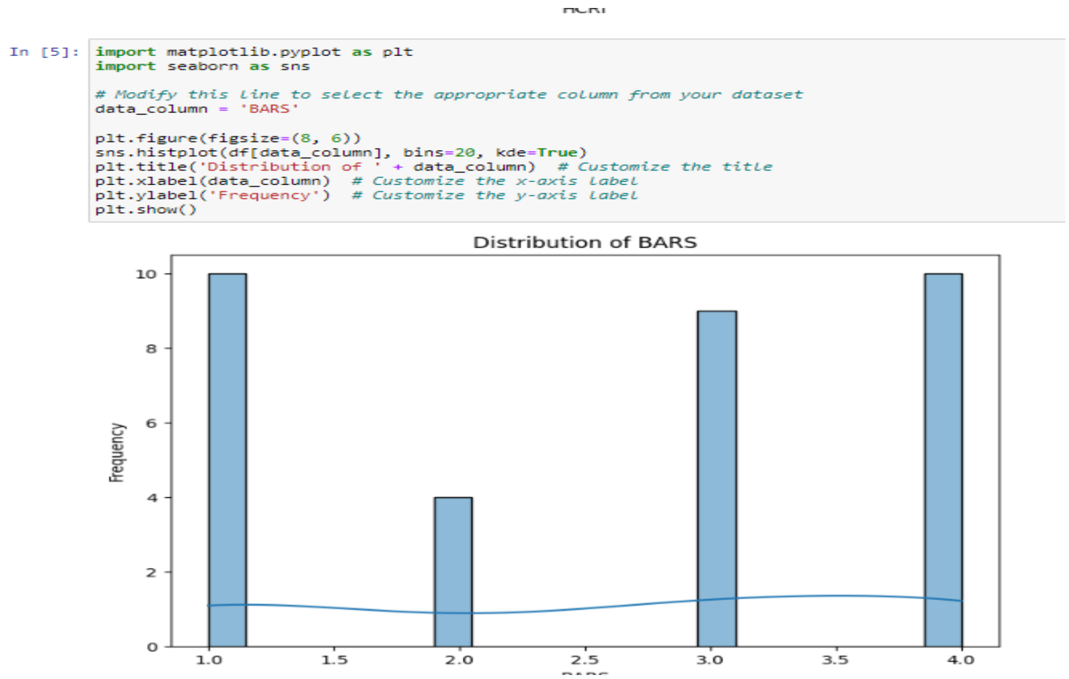
	Experience	1	2	3	4	5	1.1	2.1	3.1	4.1	...	1.4	2.4	3.4	4.4	5.4	1.5	2.5	3.5	4.5	5.5
1	15	1	5	2	1	3	1	3	5	3	...	2	3	2	1	1	1	1	5	4	4
2	7	4	2	1	4	3	4	2	5	1	...	5	2	4	3	5	3	2	2	4	1
3	16	5	3	5	2	2	1	5	5	4	...	1	2	3	3	5	1	4	1	4	4
4	4	4	1	1	5	5	1	4	4	3	...	3	1	5	5	5	1	5	3	1	2
5	13	3	1	4	1	2	5	3	4	3	...	2	2	2	3	3	5	5	2	1	5
6	17	5	5	1	3	3	5	4	2	4	...	4	5	1	1	2	1	3	2	2	5
7	17	3	2	3	5	4	1	3	2	4	...	3	2	3	4	1	4	3	2	3	4
8	2	2	4	5	3	1	1	3	1	2	...	2	5	2	3	4	1	3	2	3	5
9	17	5	2	3	3	4	4	5	5	5	...	2	1	1	1	2	1	2	4	4	1
10	7	4	1	1	1	5	1	5	3	2	...	2	4	2	4	4	1	3	2	1	5



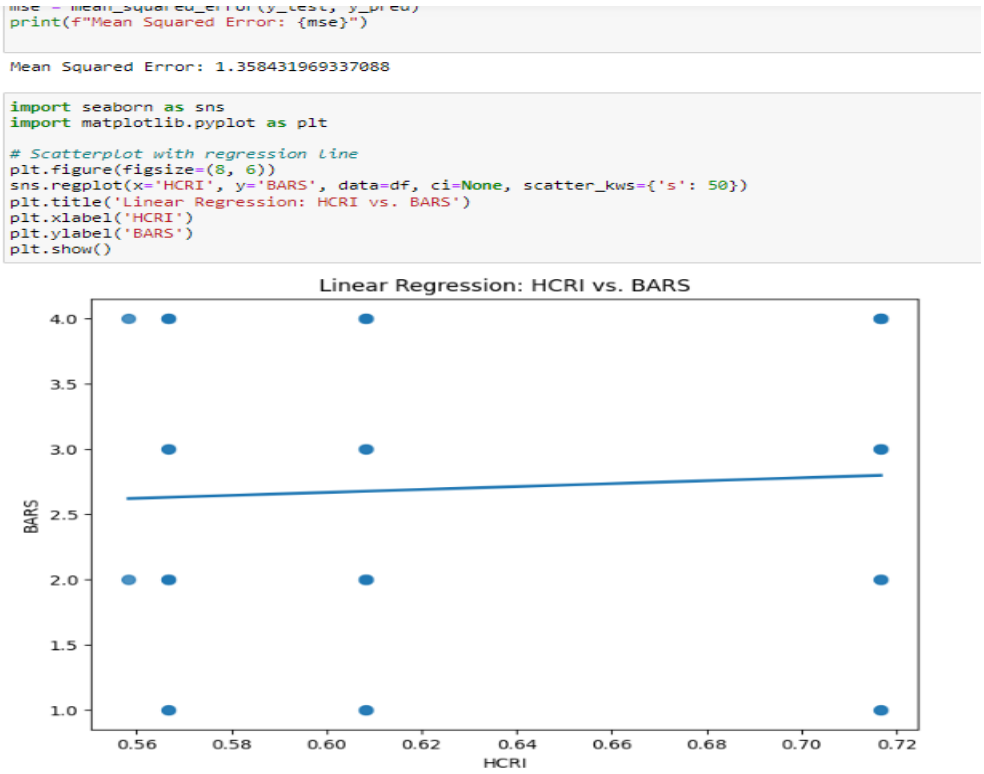
```
In [4]: import matplotlib.pyplot as plt
        import seaborn as sns

        # Modify these column names to match your dataset
        competency_column = 'HCRI'
        score_column = 'BARS'

        plt.figure(figsize=(10, 6))
        sns.barplot(x=competency_column, y=score_column, data=df)
        plt.title('Competency Factors')
        plt.xticks(rotation=90)  # Rotate x-axis labels if needed
        plt.xlabel('X Factor')  # Customize x-axis label
        plt.ylabel('Y Factor')  # Customize y-axis label
        plt.show()
```



```
[n [6]: import matplotlib.pyplot as plt
        import seaborn as sns

        # Modify this line to select the appropriate column from your dataset
        data_column = 'HCRI'

        plt.figure(figsize=(8, 6))
        sns.histplot(df[data_column], bins=20, kde=True)
        plt.title('Distribution of ' + data_column)  # Customize the title
        plt.xlabel(data_column)  # Customize the x-axis label
        plt.ylabel('Frequency')  # Customize the y-axis label
        plt.show()
```



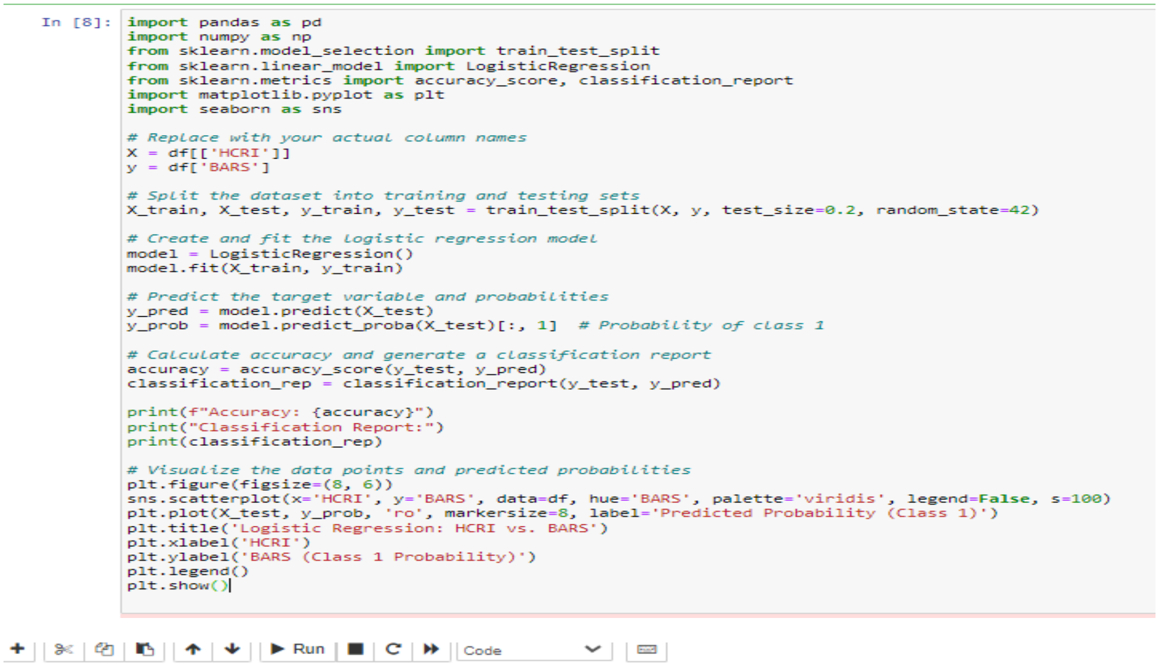
```
In [5]: import matplotlib.pyplot as plt
        import seaborn as sns

        # Modify this line to select the appropriate column from your dataset
        data_column = 'BARS'

        plt.figure(figsize=(8, 6))
        sns.histplot(df[data_column], bins=20, kde=True)
        plt.title('Distribution of ' + data_column)  # Customize the title
        plt.xlabel(data_column)  # Customize the x-axis label
        plt.ylabel('Frequency')  # Customize the y-axis label
        plt.show()
```



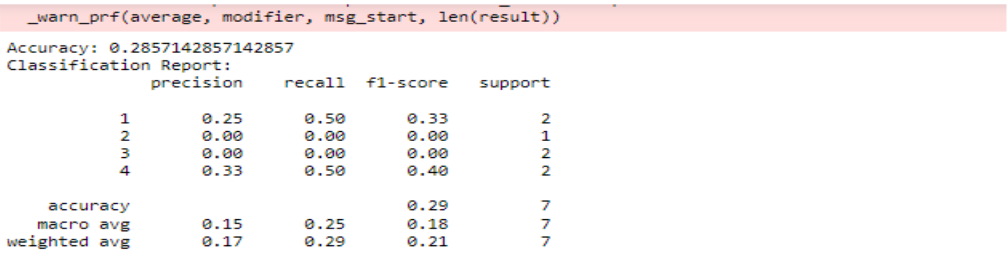
```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 1.358431969337088
```

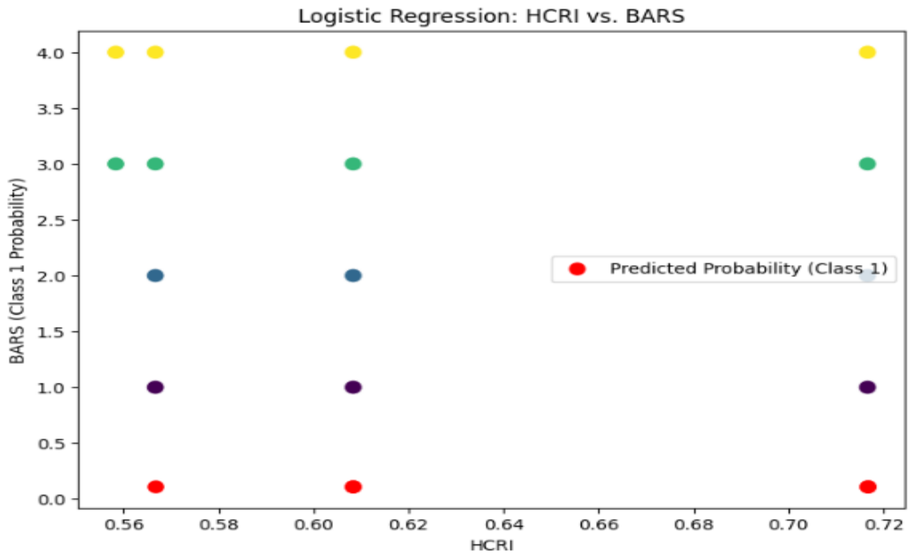
```
import seaborn as sns
import matplotlib.pyplot as plt

# Scatterplot with regression line
plt.figure(figsize=(8, 6))
sns.regplot(x='HCRI', y='BARS', data=df, ci=None, scatter_kws={'s': 50})
plt.title('Linear Regression: HCRI vs. BARS')
plt.xlabel('HCRI')
plt.ylabel('BARS')
plt.show()
```



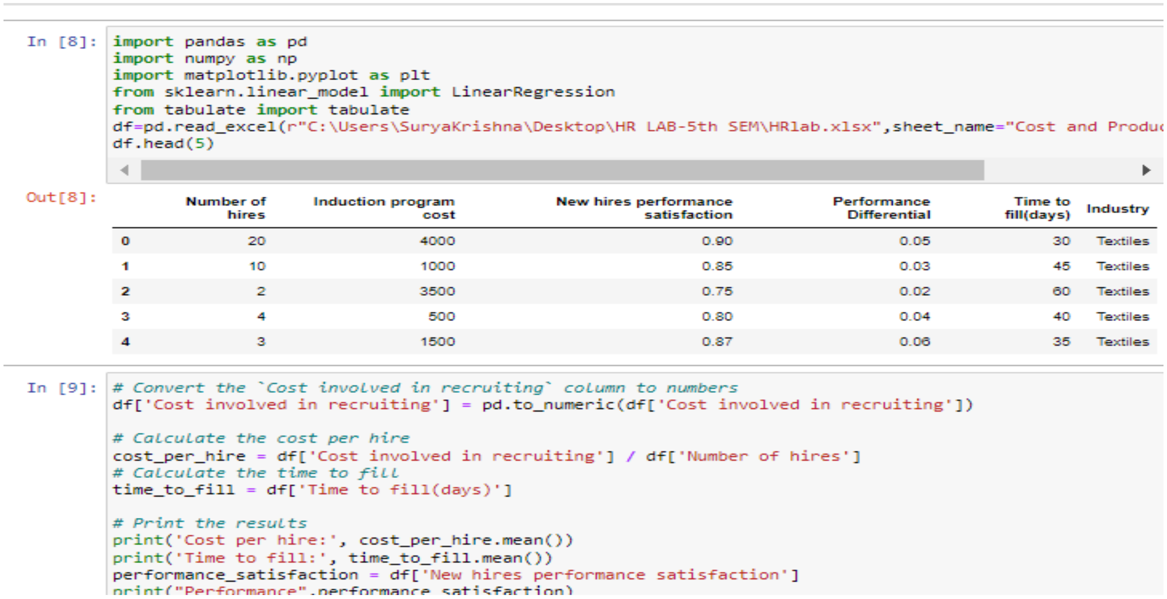
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Replace with your actual column names
X = df[['HCRI']]
y = df['BARS']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict the target variable and probabilities
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]  # Probability of class 1

# Calculate accuracy and generate a classification report
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_rep)

# Visualize the data points and predicted probabilities
plt.figure(figsize=(8, 6))
sns.scatterplot(x='HCRI', y='BARS', data=df, hue='BARS', palette='viridis', legend=False, s=100)
plt.plot(X_test, y_prob, 'ro', markersize=8, label='Predicted Probability (Class 1)')
plt.title('Logistic Regression: HCRI vs. BARS')
plt.xlabel('HCRI')
plt.ylabel('BARS (Class 1 Probability)')
plt.legend()
plt.show()
```



```
  _warn_prf(average, modifier, msg_start, len(result))

Accuracy: 0.2857142857142857
Classification Report:
              precision    recall  f1-score   support

           1       0.25      0.50      0.33         2
           2       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         2
           4       0.33      0.50      0.40         2

    accuracy                           0.29         7
   macro avg       0.15      0.25      0.18         7
weighted avg       0.17      0.29      0.21         7
```



Cost And Productivity Analysis:

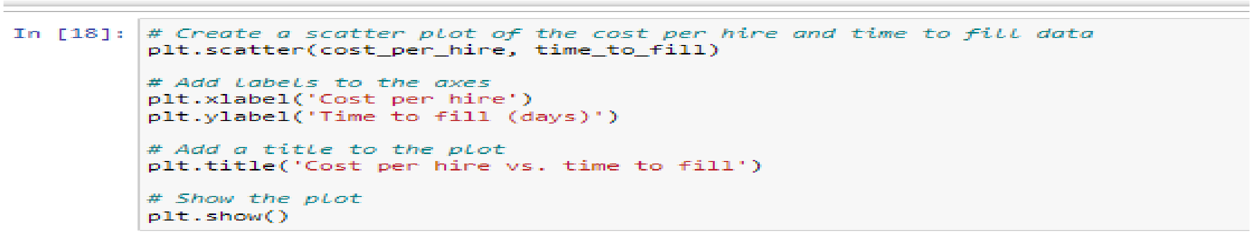


In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from tabulate import tabulate
df=pd.read_excel(r"C:\Users\SuryaKrishna\Desktop\HR LAB-5th SEM\HRlab.xlsx",sheet_name="Cost and Produc
df.head(5)
```

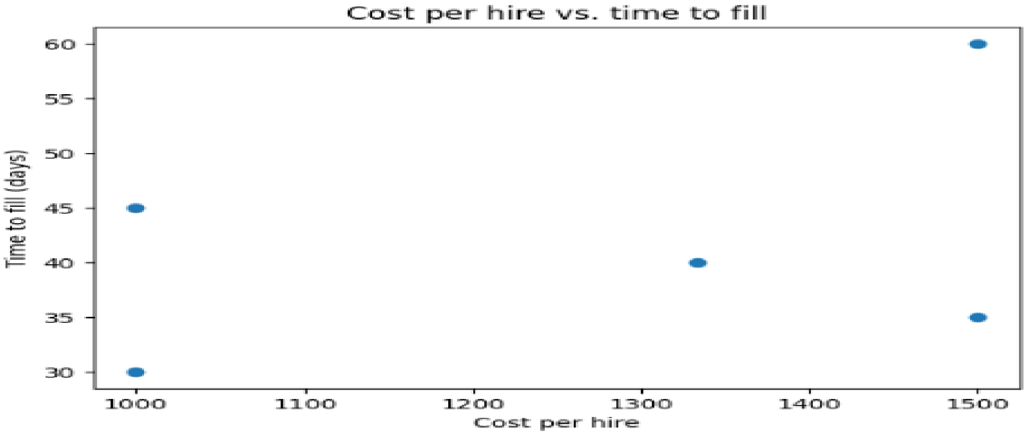
Out[8]:

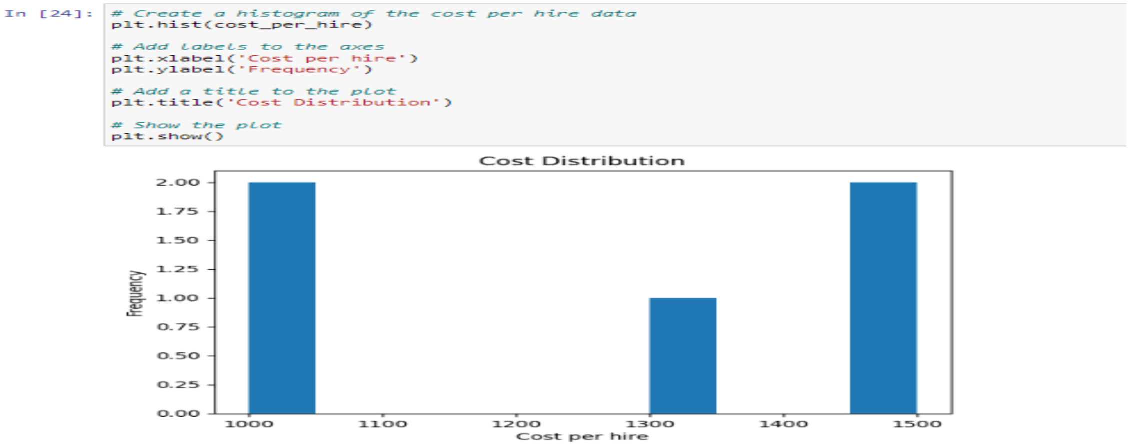
	Number of hires	Induction program cost	New hires performance satisfaction	Performance Differential	Time to fill(days)	Industry
0	20	4000	0.90	0.05	30	Textiles
1	10	1000	0.85	0.03	45	Textiles
2	2	3500	0.75	0.02	60	Textiles
3	4	500	0.80	0.04	40	Textiles
4	3	1500	0.87	0.06	35	Textiles

In [9]:
```
# Convert the `Cost involved in recruiting` column to numbers
df['Cost involved in recruiting'] = pd.to_numeric(df['Cost involved in recruiting'])

# Calculate the cost per hire
cost_per_hire = df['Cost involved in recruiting'] / df['Number of hires']
# Calculate the time to fill
time_to_fill = df['Time to fill(days)']

# Print the results
print('Cost per hire:', cost_per_hire.mean())
print('Time to fill:', time_to_fill.mean())
performance_satisfaction = df['New hires performance satisfaction']
print("Performance",performance_satisfaction)
```



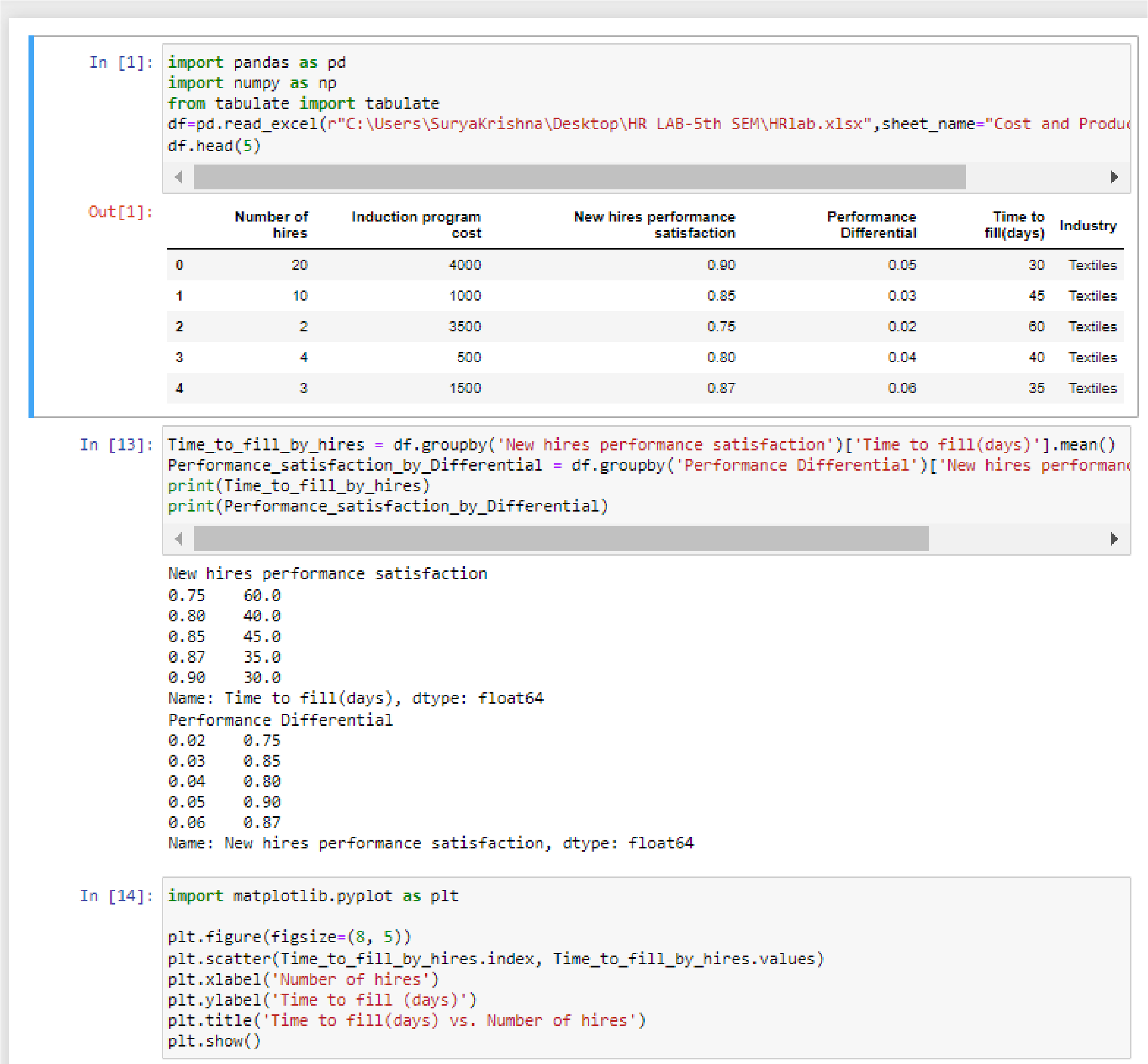
In [18]:
```
# Create a scatter plot of the cost per hire and time to fill data
plt.scatter(cost_per_hire, time_to_fill)

# Add labels to the axes
plt.xlabel('Cost per hire')
plt.ylabel('Time to fill (days)')

# Add a title to the plot
plt.title('Cost per hire vs. time to fill')

# Show the plot
plt.show()
```





```
In [24]: # Create a histogram of the cost per hire data
         plt.hist(cost_per_hire)

         # Add labels to the axes
         plt.xlabel('Cost per hire')
         plt.ylabel('Frequency')

         # Add a title to the plot
         plt.title('Cost Distribution')

         # Show the plot
         plt.show()
```

Recruitment Analysis:



```
In [1]: import pandas as pd
        import numpy as np
        from tabulate import tabulate
        df=pd.read_excel(r"C:\Users\SuryaKrishna\Desktop\HR LAB-5th SEM\HRlab.xlsx",sheet_name="Cost and Produc
        df.head(5)
```

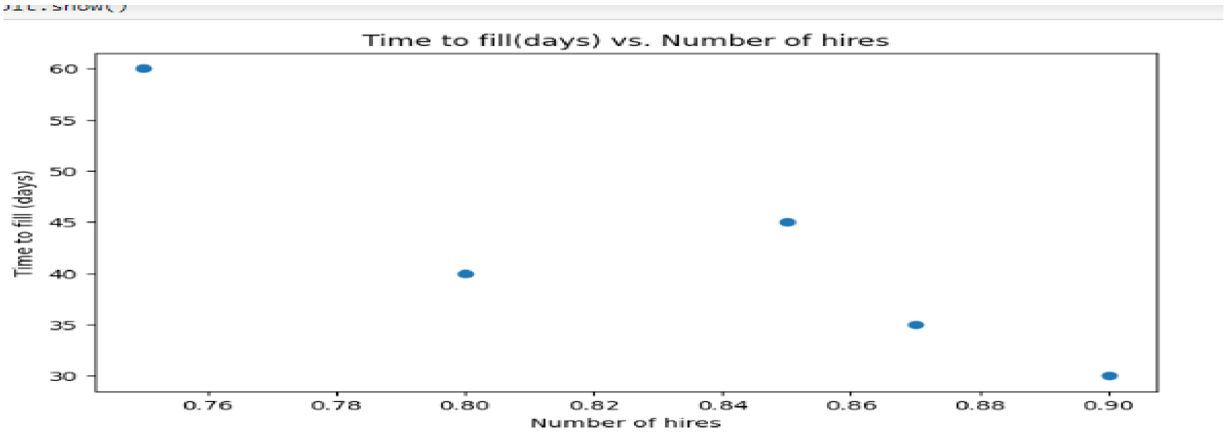
Out[1]:

	Number of hires	Induction program cost	New hires performance satisfaction	Performance Differential	Time to fill(days)	Industry
0	20	4000	0.90	0.05	30	Textiles
1	10	1000	0.85	0.03	45	Textiles
2	2	3500	0.75	0.02	60	Textiles
3	4	500	0.80	0.04	40	Textiles
4	3	1500	0.87	0.06	35	Textiles

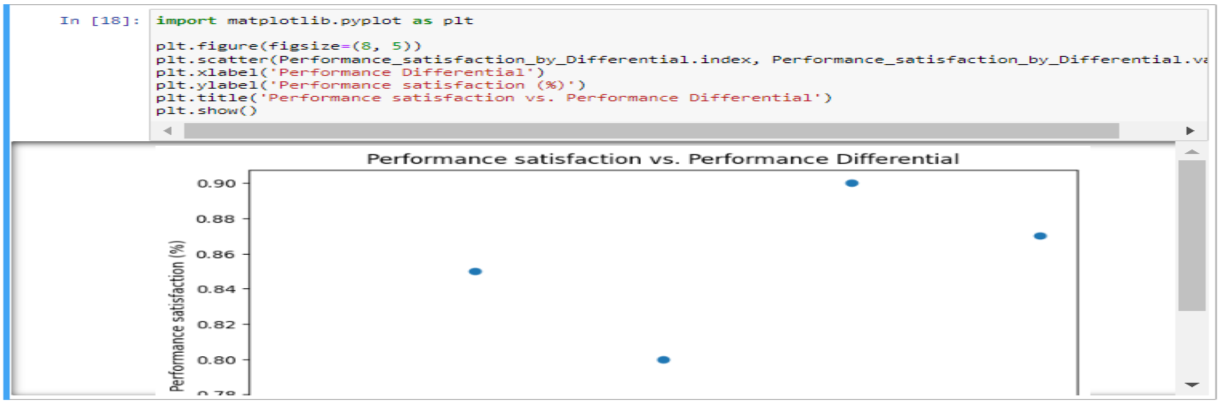
```
In [13]: Time_to_fill_by_hires = df.groupby('New hires performance satisfaction')['Time to fill(days)'].mean()
         Performance_satisfaction_by_Differential = df.groupby('Performance Differential')['New hires performanc
         print(Time_to_fill_by_hires)
         print(Performance_satisfaction_by_Differential)
```

```
New hires performance satisfaction
0.75    60.0
0.80    40.0
0.85    45.0
0.87    35.0
0.90    30.0
Name: Time to fill(days), dtype: float64
Performance Differential
0.02    0.75
0.03    0.85
0.04    0.80
0.05    0.90
0.06    0.87
Name: New hires performance satisfaction, dtype: float64
```

```
In [14]: import matplotlib.pyplot as plt

         plt.figure(figsize=(8, 5))
         plt.scatter(Time_to_fill_by_hires.index, Time_to_fill_by_hires.values)
         plt.xlabel('Number of hires')
         plt.ylabel('Time to fill (days)')
         plt.title('Time to fill(days) vs. Number of hires')
         plt.show()
```

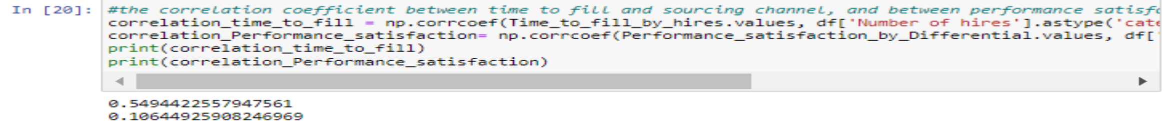


```
plt.show()
```

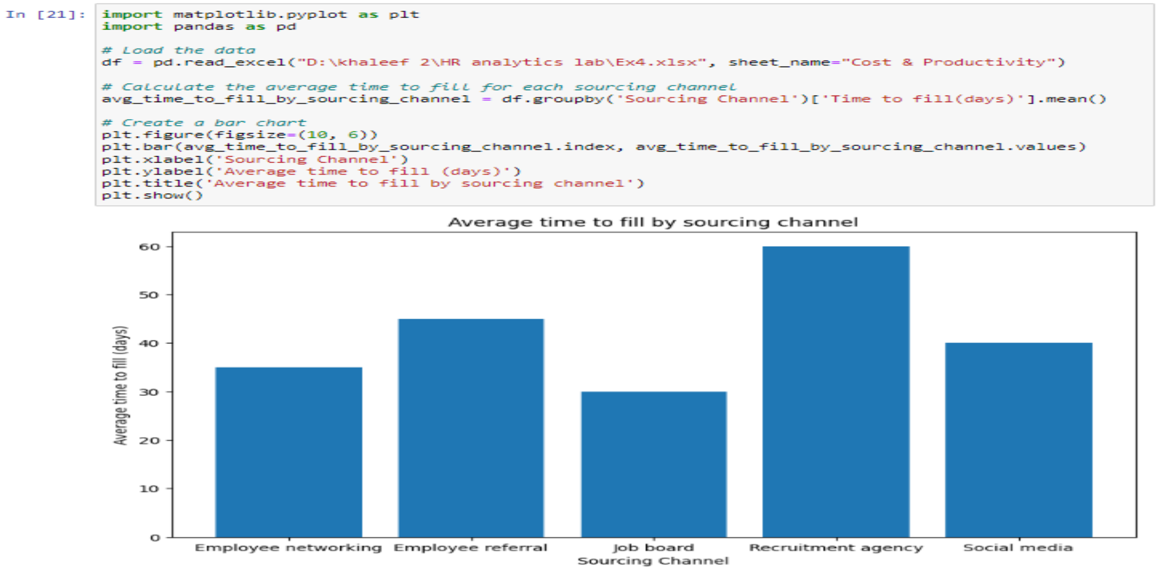


```
In [18]: import matplotlib.pyplot as plt

         plt.figure(figsize=(8, 5))
         plt.scatter(Performance_satisfaction_by_Differential.index, Performance_satisfaction_by_Differential.va
         plt.xlabel('Performance Differential')
         plt.ylabel('Performance satisfaction (%)')
         plt.title('Performance satisfaction vs. Performance Differential')
         plt.show()
```

Performance satisfaction vs. Performance Differential
Performance satisfaction (%)
0.90
0.88
0.86
0.84
0.82
0.80
0.78

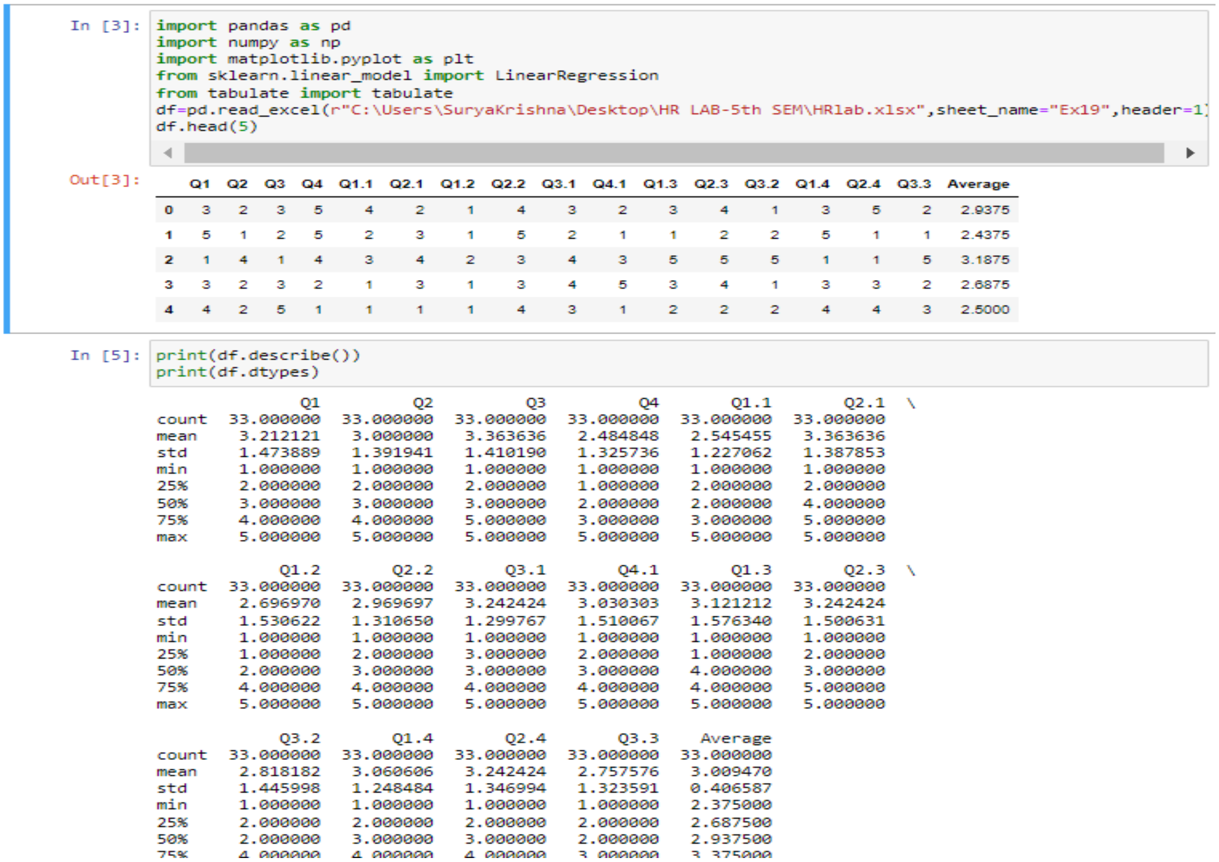


```
In [20]: #the correlation coefficient between time to fill and sourcing channel, and between performance satisfa
         correlation_time_to_fill = np.corrcoef(Time_to_fill_by_hires.values, df['Number of hires'].astype('cate
         correlation_Performance_satisfaction= np.corrcoef(Performance_satisfaction_by_Differential.values, df['
         print(correlation_time_to_fill)
         print(correlation_Performance_satisfaction)

         0.5494422557947561
         0.10644925908246969
```



```
In [21]: import matplotlib.pyplot as plt
         import pandas as pd

         # Load the data
         df = pd.read_excel("D:\khaleef 2\HR analytics lab\Ex4.xlsx", sheet_name="Cost & Productivity")

         # Calculate the average time to fill for each sourcing channel
         avg_time_to_fill_by_sourcing_channel = df.groupby('Sourcing Channel')['Time to fill(days)'].mean()

         # Create a bar chart
         plt.figure(figsize=(10, 6))
         plt.bar(avg_time_to_fill_by_sourcing_channel.index, avg_time_to_fill_by_sourcing_channel.values)
         plt.xlabel('Sourcing Channel')
         plt.ylabel('Average time to fill (days)')
         plt.title('Average time to fill by sourcing channel')
         plt.show()
```

Training and Development Analysis:



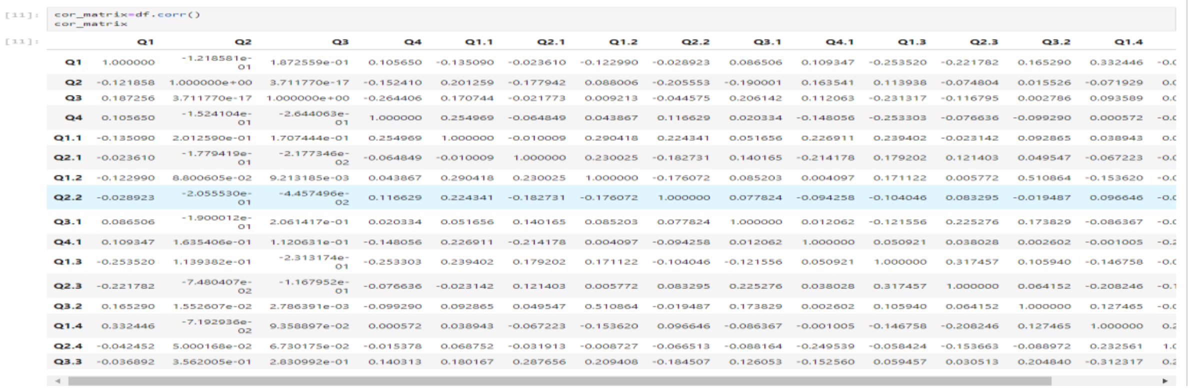
In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from tabulate import tabulate
df=pd.read_excel(r"C:\Users\SuryaKrishna\Desktop\HR LAB-5th SEM\HRlab.xlsx",sheet_name="Ex19",header=1)
df.head(5)
```

Out[3]:

	Q1	Q2	Q3	Q4	Q1.1	Q2.1	Q1.2	Q2.2	Q3.1	Q4.1	Q1.3	Q2.3	Q3.2	Q1.4	Q2.4	Q3.3	Average
0	3	2	3	5	4	2	1	4	3	2	3	4	1	3	5	2	2.9375
1	5	1	2	5	2	3	1	5	2	1	1	2	2	5	1	1	2.4375
2	1	4	1	4	3	4	2	3	4	3	5	5	5	1	1	5	3.1875
3	3	2	3	2	1	3	1	3	4	5	3	4	1	3	3	2	2.6875
4	4	2	5	1	1	1	1	4	3	1	2	2	2	4	4	3	2.5000

In [5]:
```
print(df.describe())
print(df.dtypes)
```

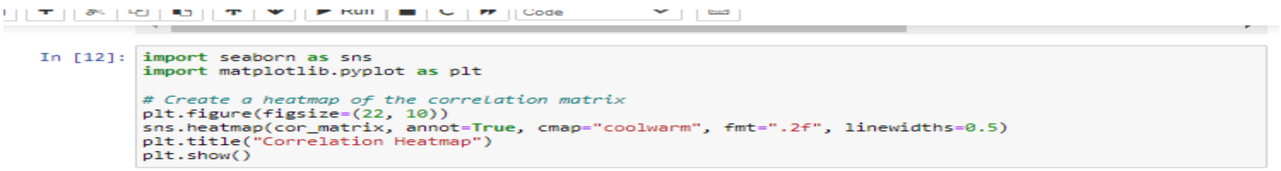
```
              Q1         Q2         Q3         Q4       Q1.1       Q2.1  \
count  33.000000  33.000000  33.000000  33.000000  33.000000  33.000000   
mean    3.212121   3.000000   3.363636   2.484848   2.545455   3.363636   
std     1.473889   1.391941   1.410190   1.325736   1.227062   1.387853   
min     1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   
25%     2.000000   2.000000   2.000000   1.000000   2.000000   2.000000   
50%     3.000000   3.000000   3.000000   2.000000   2.000000   4.000000   
75%     4.000000   4.000000   5.000000   3.000000   3.000000   5.000000   
max     5.000000   5.000000   5.000000   5.000000   5.000000   5.000000   

            Q1.2       Q2.2       Q3.1       Q4.1       Q1.3       Q2.3  \
count  33.000000  33.000000  33.000000  33.000000  33.000000  33.000000   
mean    2.696970   2.969697   3.242424   3.030303   3.121212   3.242424   
std     1.530622   1.310650   1.299767   1.510067   1.576340   1.500631   
min     1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   
25%     1.000000   2.000000   3.000000   2.000000   1.000000   2.000000   
50%     2.000000   3.000000   3.000000   3.000000   4.000000   3.000000   
75%     4.000000   4.000000   4.000000   4.000000   4.000000   5.000000   
max     5.000000   5.000000   5.000000   5.000000   5.000000   5.000000   

            Q3.2       Q1.4       Q2.4       Q3.3    Average  
count  33.000000  33.000000  33.000000  33.000000  33.000000  
mean    2.818182   3.060606   3.242424   2.757576   3.009470  
std     1.445998   1.248484   1.346994   1.323591   0.406587  
min     1.000000   1.000000   1.000000   1.000000   2.375000  
25%     2.000000   2.000000   2.000000   2.000000   2.687500  
50%     2.000000   3.000000   3.000000   2.000000   2.937500  
75%     4.000000   4.000000   4.000000   3.000000   3.375000  
```



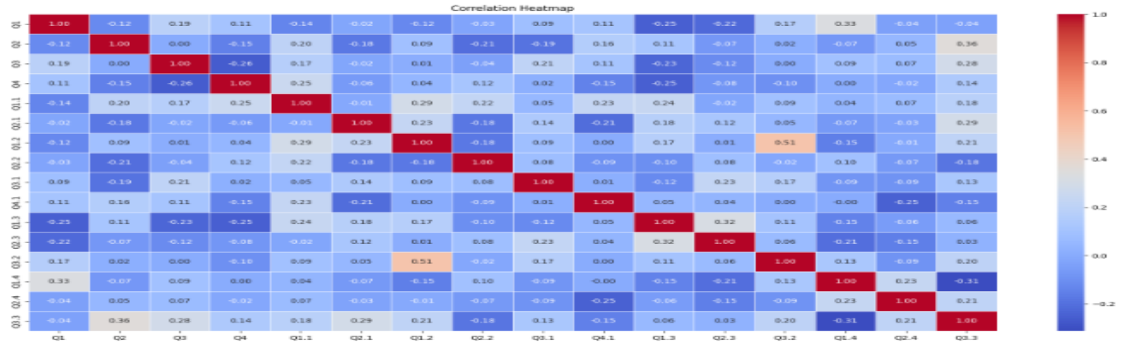
[11]:
```
cor_matrix=df.corr()
cor_matrix
```

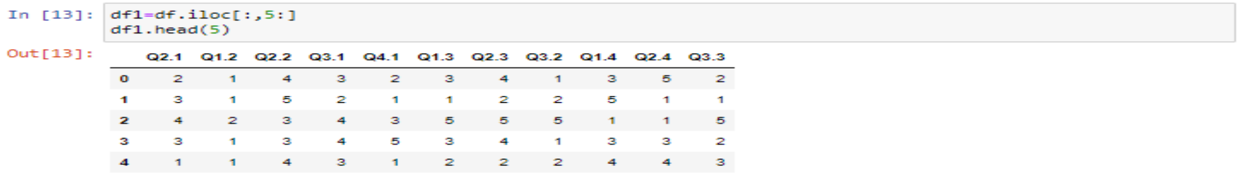
[11]:

	Q1	Q2	Q3	Q4	Q1.1	Q2.1	Q1.2	Q2.2	Q3.1	Q4.1	Q1.3	Q2.3	Q3.2	Q1.4	
Q1	1.000000	-1.218581e-01	1.872559e-01	0.105650	-0.135090	-0.023610	-0.122990	-0.028923	0.086506	0.109347	-0.253520	-0.221782	0.165290	0.332446	-0.0
Q2	-0.121858	1.000000e+00	3.711770e-17	-0.152410	0.201259	-0.177942	0.088006	-0.205553	-0.190001	0.163541	0.113938	-0.074804	0.015526	-0.071929	0.0
Q3	0.187256	3.711770e-17	1.000000e+00	-0.264406	0.170744	-0.021773	0.009213	-0.044575	0.206142	0.112063	-0.231317	-0.116795	0.002786	0.093589	0.0
Q4	0.105650	-1.524104e-01	-2.644063e-01	1.000000	0.254969	-0.064849	0.043867	0.116629	0.020334	-0.148056	-0.253303	-0.076636	-0.099290	0.000572	-0.0
Q1.1	-0.135090	2.012590e-01	1.707444e-01	0.254969	1.000000	-0.010009	0.290418	0.224341	0.051656	0.226911	0.239402	-0.023142	0.092865	0.038943	0.0
Q2.1	-0.023610	-1.779419e-01	-2.177346e-02	-0.064849	-0.010009	1.000000	0.230025	-0.182731	0.140165	-0.214178	0.179202	0.121403	0.049547	-0.067223	-0.0
Q1.2	-0.122990	8.800605e-02	9.213185e-03	0.043867	0.290418	0.230025	1.000000	-0.176072	0.085203	0.004097	0.171122	0.005772	0.510864	-0.153620	-0.0
Q2.2	-0.028923	-2.055530e-01	-4.457496e-02	0.116629	0.224341	-0.182731	-0.176072	1.000000	0.077824	-0.094258	-0.104046	0.083295	-0.019487	0.096646	-0.0
Q3.1	0.086506	-1.900012e-01	2.061417e-01	0.020334	0.051656	0.140165	0.085203	0.077824	1.000000	0.012062	-0.121556	0.225276	0.173829	-0.086367	-0.0
Q4.1	0.109347	1.635406e-01	1.120631e-01	-0.148056	0.226911	-0.214178	0.004097	-0.094258	0.012062	1.000000	0.050921	0.038028	0.002602	-0.001005	-0.2
Q1.3	-0.253520	1.139382e-01	-2.313174e-01	-0.253303	0.239402	0.179202	0.171122	-0.104046	-0.121556	0.050921	1.000000	0.317457	0.105940	-0.146758	-0.0
Q2.3	-0.221782	-7.480407e-02	-1.167952e-01	-0.076636	-0.023142	0.121403	0.005772	0.083295	0.225276	0.038028	0.317457	1.000000	0.064152	-0.208246	-0.1
Q3.2	0.165290	1.552607e-02	2.786391e-03	-0.099290	0.092865	0.049547	0.510864	-0.019487	0.173829	0.002602	0.105940	0.064152	1.000000	0.127465	-0.0
Q1.4	0.332446	-7.192936e-02	9.358897e-02	0.000572	0.038943	-0.067223	-0.153620	0.096646	-0.086367	-0.001005	-0.146758	-0.208246	0.127465	1.000000	0.2
Q2.4	-0.042452	5.000168e-02	6.730175e-02	-0.015378	0.068752	-0.031913	-0.008727	-0.066513	-0.088164	-0.249539	-0.058424	-0.153663	-0.088972	0.232561	1.0
Q3.3	-0.036892	3.562005e-01	2.830992e-01	0.140313	0.180167	0.287656	0.209408	-0.184507	0.126053	-0.152560	0.059457	0.030513	0.204840	-0.312317	0.2



In [12]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a heatmap of the correlation matrix
plt.figure(figsize=(22, 10))
sns.heatmap(cor_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```

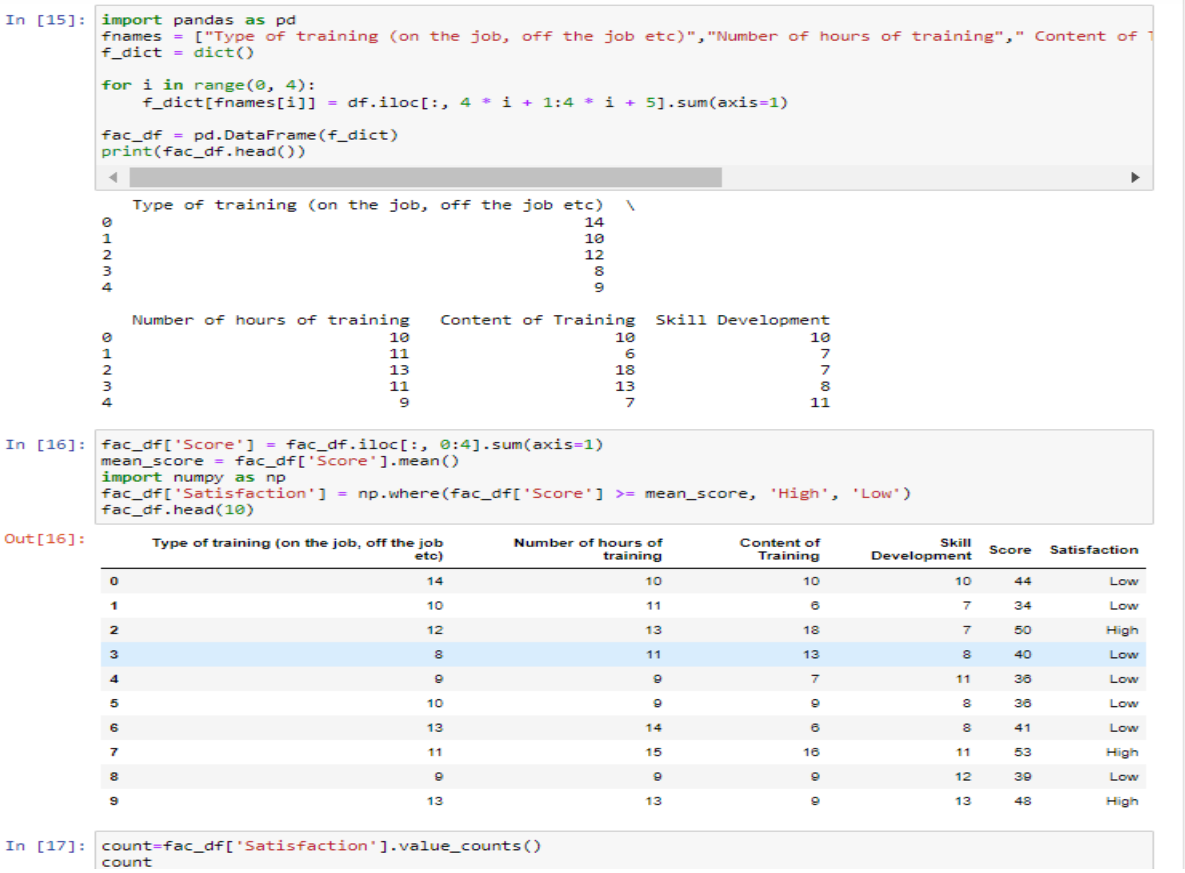




In [13]:
```
df1=df.iloc[:,5:]
df1.head(5)
```

Out[13]:

	Q2.1	Q1.2	Q2.2	Q3.1	Q4.1	Q1.3	Q2.3	Q3.2	Q1.4	Q2.4	Q3.3
0	2	1	4	3	2	3	4	1	3	5	2
1	3	1	5	2	1	1	2	2	5	1	1
2	4	2	3	4	3	5	5	5	1	1	5
3	3	1	3	4	5	3	4	1	3	3	2
4	1	1	4	3	1	2	2	2	4	4	3



In [15]:
```
import pandas as pd
fnames = ["Type of training (on the job, off the job etc)","Number of hours of training"," Content of T
f_dict = dict()

for i in range(0, 4):
    f_dict[fnames[i]] = df.iloc[:, 4 * i + 1:4 * i + 5].sum(axis=1)

fac_df = pd.DataFrame(f_dict)
print(fac_df.head())
```

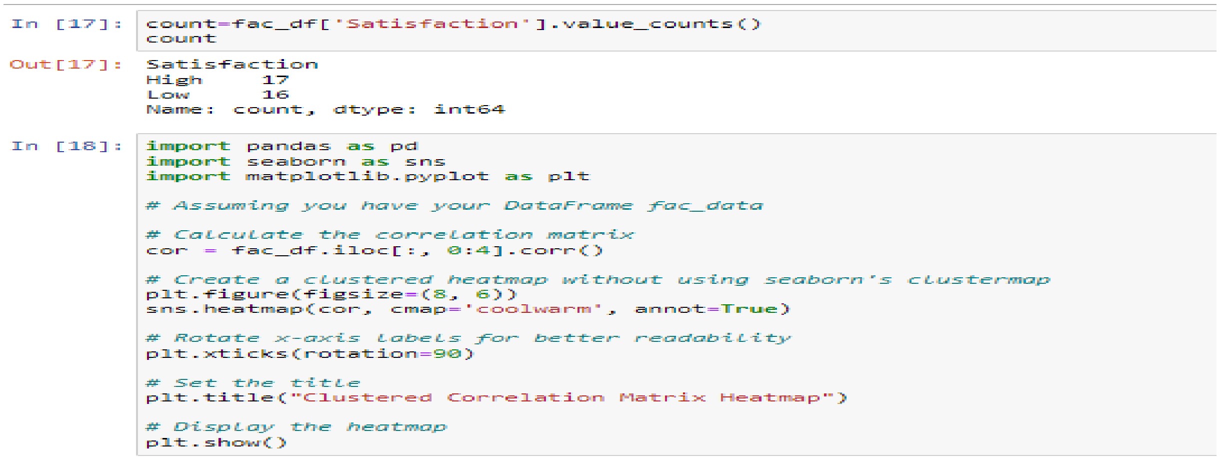
```
   Type of training (on the job, off the job etc)  \
0                                              14
1                                              10
2                                              12
3                                               8
4                                               9

   Number of hours of training  Content of Training  Skill Development
0                           10                   10                 10
1                           11                    6                  7
2                           13                   18                  7
3                           11                   13                  8
4                            9                    7                 11
```

In [16]:
```
fac_df['Score'] = fac_df.iloc[:, 0:4].sum(axis=1)
mean_score = fac_df['Score'].mean()
import numpy as np
fac_df['Satisfaction'] = np.where(fac_df['Score'] >= mean_score, 'High', 'Low')
fac_df.head(10)
```

Out[16]:

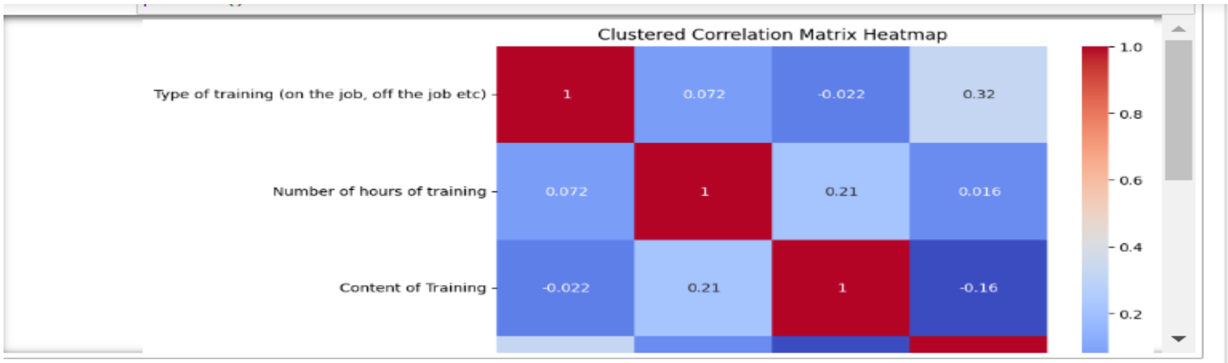
	Type of training (on the job, off the job etc)	Number of hours of training	Content of Training	Skill Development	Score	Satisfaction
0	14	10	10	10	44	Low
1	10	11	6	7	34	Low
2	12	13	18	7	50	High
3	8	11	13	8	40	Low
4	9	9	7	11	36	Low
5	10	9	9	8	36	Low
6	13	14	6	8	41	Low
7	11	15	16	11	53	High
8	9	9	9	12	39	Low
9	13	13	9	13	48	High

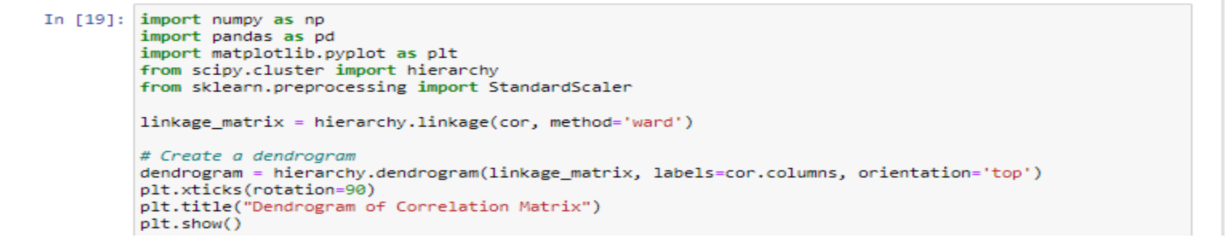
In [17]:
```
count=fac_df['Satisfaction'].value_counts()
count
```



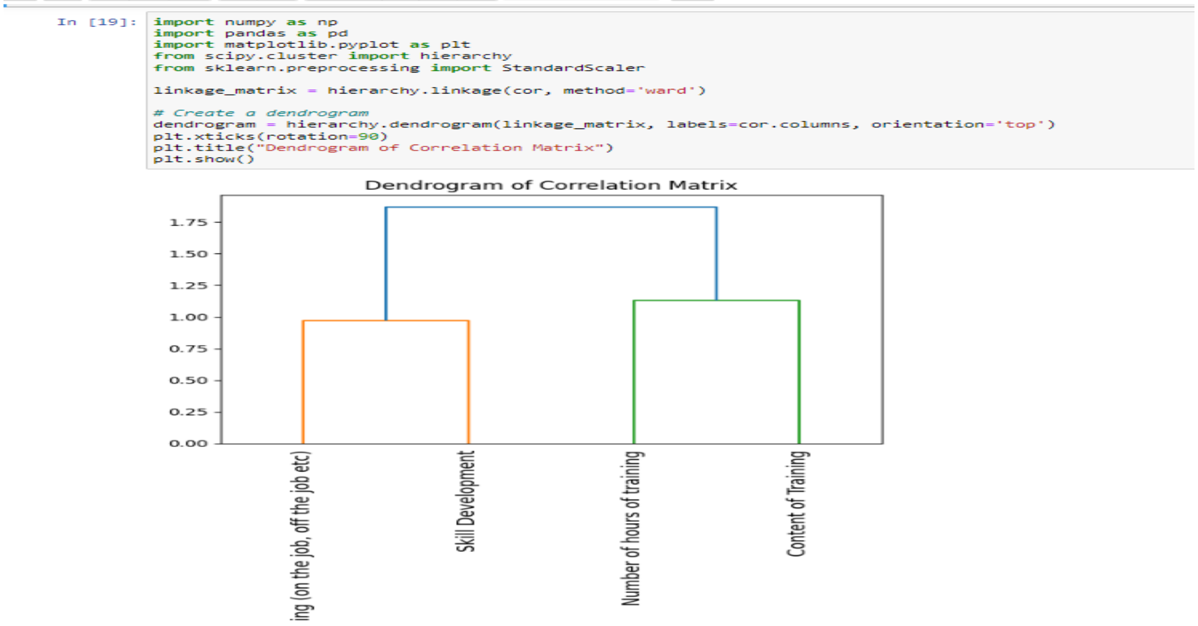
```
In [17]: count=fac_df['Satisfaction'].value_counts()
         count
Out[17]: Satisfaction
         High     17
         Low      16
         Name: count, dtype: int64
```

```
In [18]: import pandas as pd
         import seaborn as sns
         import matplotlib.pyplot as plt

         # Assuming you have your DataFrame fac_data

         # Calculate the correlation matrix
         cor = fac_df.iloc[:, 0:4].corr()

         # Create a clustered heatmap without using seaborn's clustermap
         plt.figure(figsize=(8, 6))
         sns.heatmap(cor, cmap='coolwarm', annot=True)

         # Rotate x-axis labels for better readability
         plt.xticks(rotation=90)

         # Set the title
         plt.title("Clustered Correlation Matrix Heatmap")

         # Display the heatmap
         plt.show()
```


Clustered Correlation Matrix Heatmap
Type of training (on the job, off the job etc)
1
0.072
-0.022
0.32
Number of hours of training
0.072
1
0.21
0.016
Content of Training
-0.022
0.21
1
-0.16
1.0
0.8
0.6
0.4
0.2



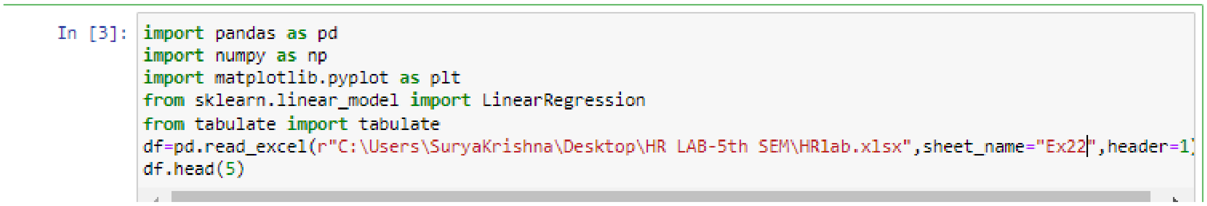
```
In [19]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         from scipy.cluster import hierarchy
         from sklearn.preprocessing import StandardScaler

         linkage_matrix = hierarchy.linkage(cor, method='ward')

         # Create a dendrogram
         dendrogram = hierarchy.dendrogram(linkage_matrix, labels=cor.columns, orientation='top')
         plt.xticks(rotation=90)
         plt.title("Dendrogram of Correlation Matrix")
         plt.show()
```



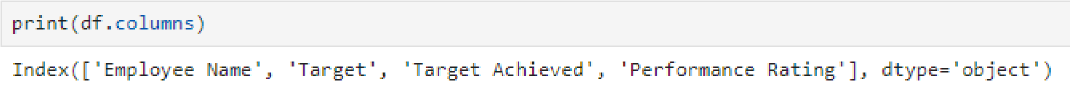
```
In [19]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         from scipy.cluster import hierarchy
         from sklearn.preprocessing import StandardScaler

         linkage_matrix = hierarchy.linkage(cor, method='ward')

         # Create a dendrogram
         dendrogram = hierarchy.dendrogram(linkage_matrix, labels=cor.columns, orientation='top')
         plt.xticks(rotation=90)
         plt.title("Dendrogram of Correlation Matrix")
         plt.show()
```

Dendrogram of Correlation Matrix
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
ing (on the job, off the job etc)
Skill Development
Number of hours of training
Content of Training

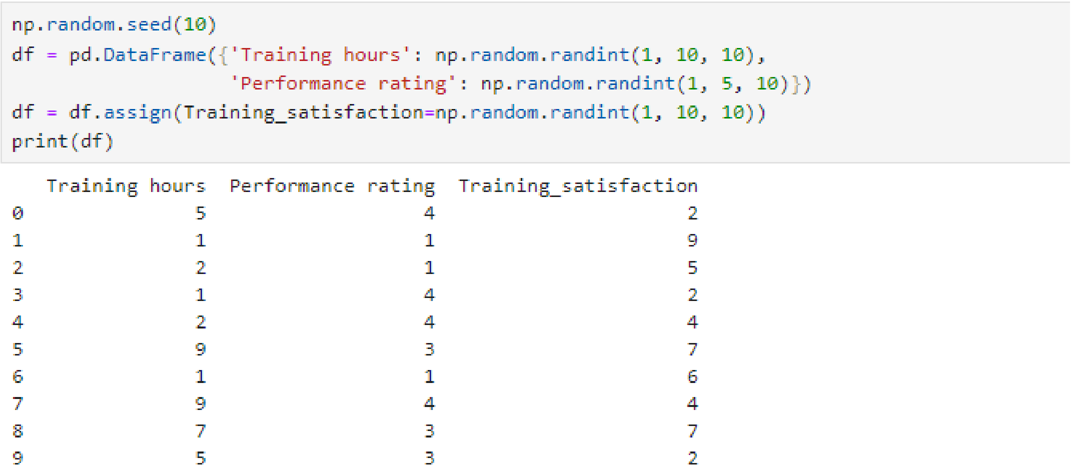
Performance Management:



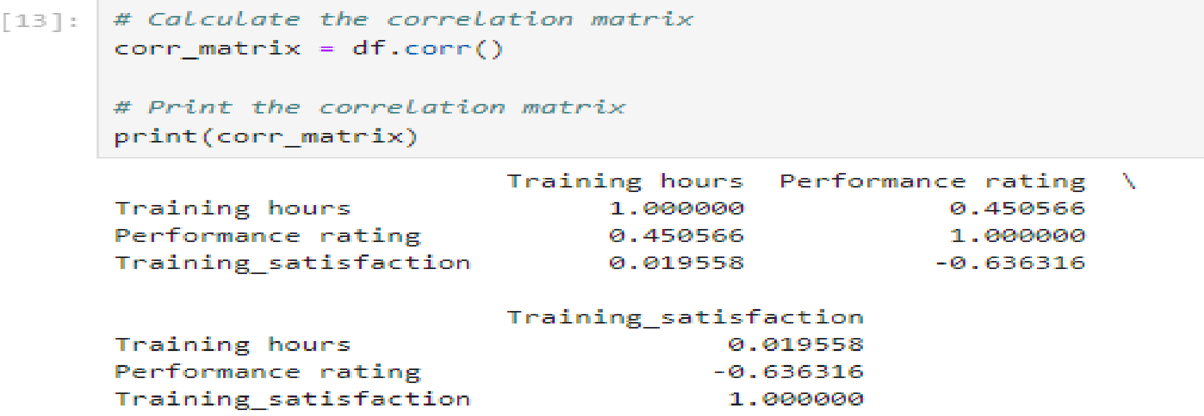
```
In [3]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.linear_model import LinearRegression
        from tabulate import tabulate
        df=pd.read_excel(r"C:\Users\SuryaKrishna\Desktop\HR LAB-5th SEM\HRlab.xlsx",sheet_name="Ex22",header=1
        df.head(5)
```



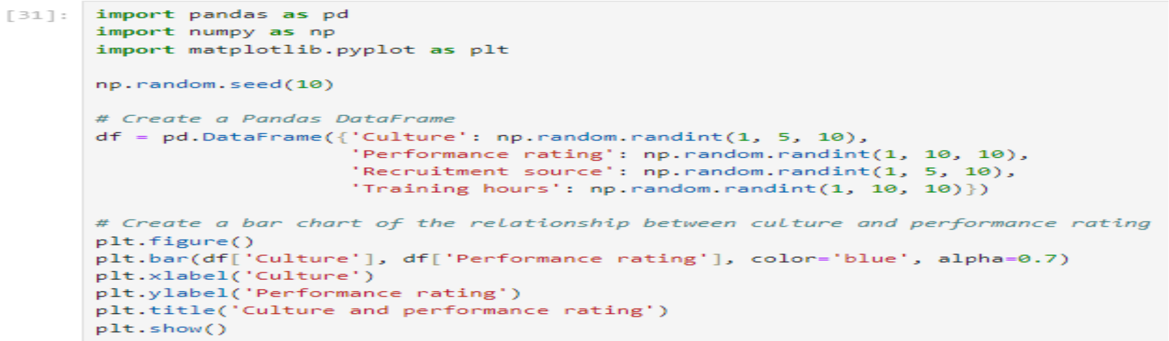
```
print(df.columns)

Index(['Employee Name', 'Target', 'Target Achieved', 'Performance Rating'], dtype='object')
```



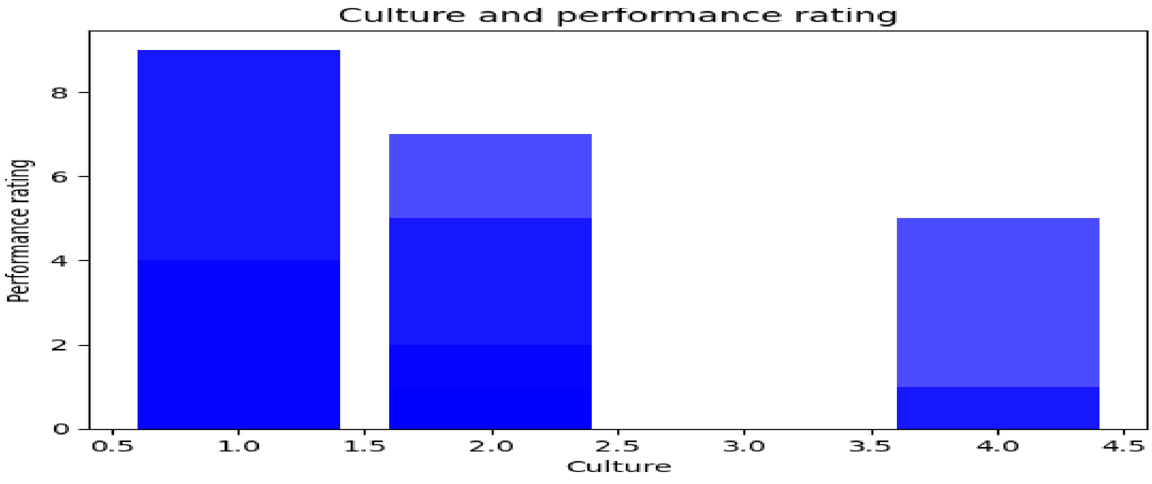
```
np.random.seed(10)
df = pd.DataFrame({'Training hours': np.random.randint(1, 10, 10),
                    'Performance rating': np.random.randint(1, 5, 10)})
df = df.assign(Training_satisfaction=np.random.randint(1, 10, 10))
print(df)

   Training hours  Performance rating  Training_satisfaction
0               5                   4                      2
1               1                   1                      9
2               2                   1                      5
3               1                   4                      2
4               2                   4                      4
5               9                   3                      7
6               1                   1                      6
7               9                   4                      4
8               7                   3                      7
9               5                   3                      2
```

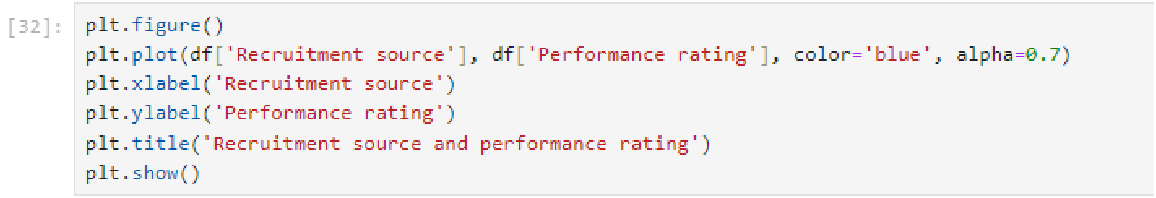


```
[13]: # Calculate the correlation matrix
      corr_matrix = df.corr()

      # Print the correlation matrix
      print(corr_matrix)

                             Training hours  Performance rating  \
      Training hours               1.000000            0.450566
      Performance rating           0.450566            1.000000
      Training_satisfaction        0.019558           -0.636316

                             Training_satisfaction
      Training hours                      0.019558
      Performance rating                 -0.636316
      Training_satisfaction               1.000000
```

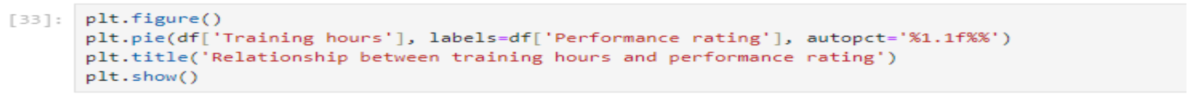


```
[31]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt

      np.random.seed(10)

      # Create a Pandas DataFrame
      df = pd.DataFrame({'Culture': np.random.randint(1, 5, 10),
                         'Performance rating': np.random.randint(1, 10, 10),
                         'Recruitment source': np.random.randint(1, 5, 10),
                         'Training hours': np.random.randint(1, 10, 10)})

      # Create a bar chart of the relationship between culture and performance rating
      plt.figure()
      plt.bar(df['Culture'], df['Performance rating'], color='blue', alpha=0.7)
      plt.xlabel('Culture')
      plt.ylabel('Performance rating')
      plt.title('Culture and performance rating')
      plt.show()
```

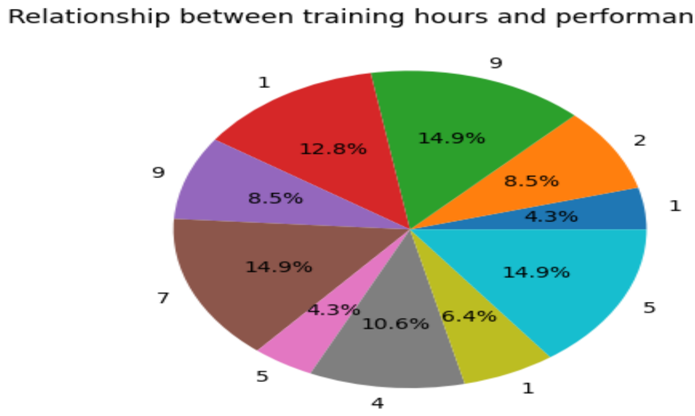




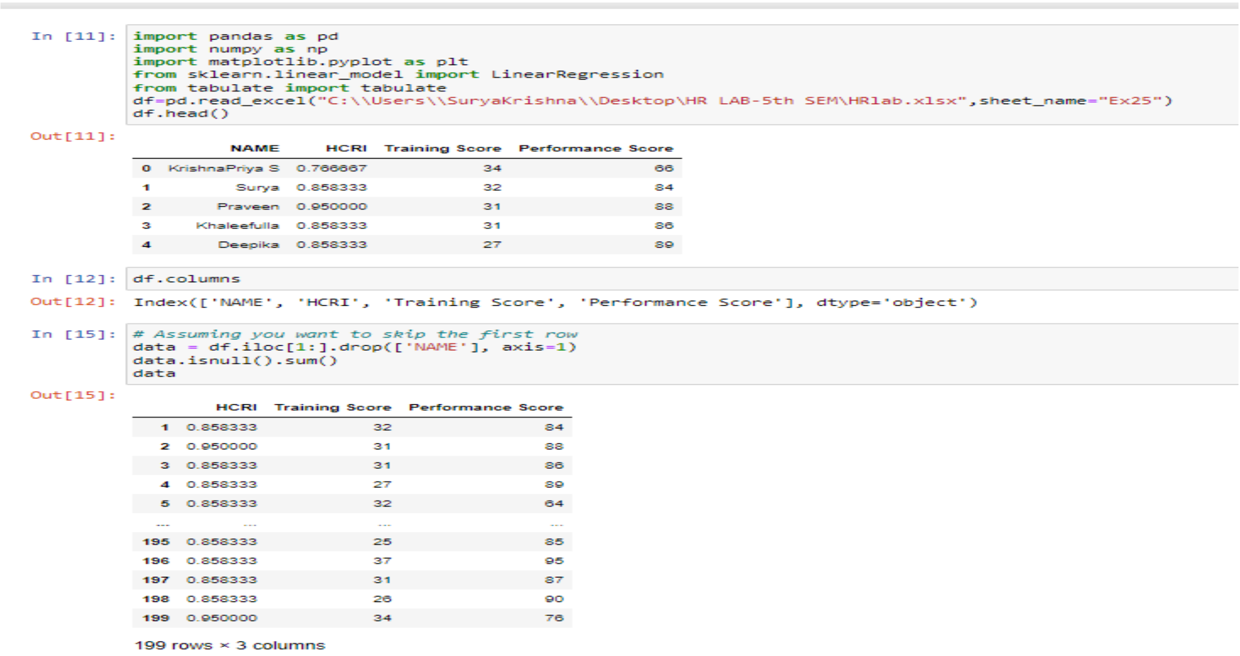
```
[32]: plt.figure()
      plt.plot(df['Recruitment source'], df['Performance rating'], color='blue', alpha=0.7)
      plt.xlabel('Recruitment source')
      plt.ylabel('Performance rating')
      plt.title('Recruitment source and performance rating')
      plt.show()
```



```
[33]: plt.figure()
      plt.pie(df['Training hours'], labels=df['Performance rating'], autopct='%1.1f%%')
      plt.title('Relationship between training hours and performance rating')
      plt.show()
```



Talent Management:



```
In [11]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.linear_model import LinearRegression
         from tabulate import tabulate
         df=pd.read_excel("C:\\Users\\SuryaKrishna\\Desktop\HR LAB-5th SEM\HRlab.xlsx",sheet_name="Ex25")
         df.head()
```

Out[11]:

	NAME	HCRI	Training Score	Performance Score
0	KrishnaPriya S	0.766667	34	66
1	Surya	0.858333	32	84
2	Praveen	0.950000	31	88
3	Khaleefulla	0.858333	31	86
4	Deepika	0.858333	27	89

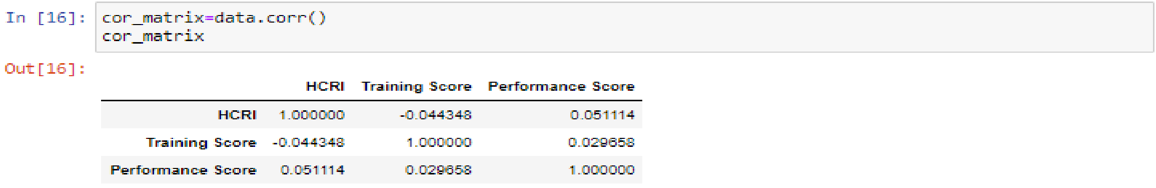
```
In [12]: df.columns
Out[12]: Index(['NAME', 'HCRI', 'Training Score', 'Performance Score'], dtype='object')
```

```
In [15]: # Assuming you want to skip the first row
         data = df.iloc[1:].drop(['NAME'], axis=1)
         data.isnull().sum()
         data
```

Out[15]:

	HCRI	Training Score	Performance Score
1	0.858333	32	84
2	0.950000	31	88
3	0.858333	31	86
4	0.858333	27	89
5	0.858333	32	64
...	...	...	...
195	0.858333	25	85
196	0.858333	37	95
197	0.858333	31	87
198	0.858333	26	90
199	0.950000	34	76

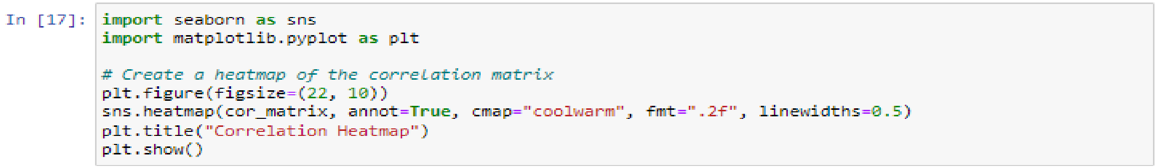
199 rows × 3 columns



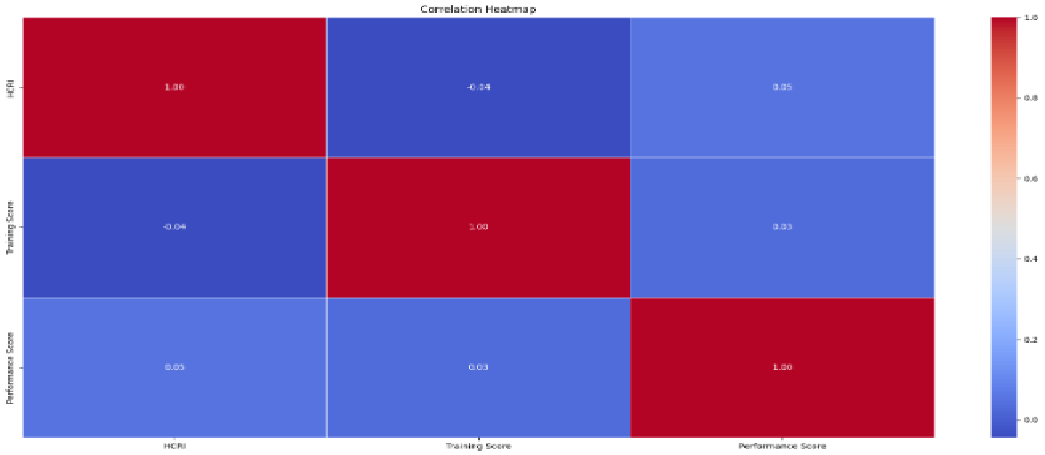
```
In [16]: cor_matrix=data.corr()
         cor_matrix
```

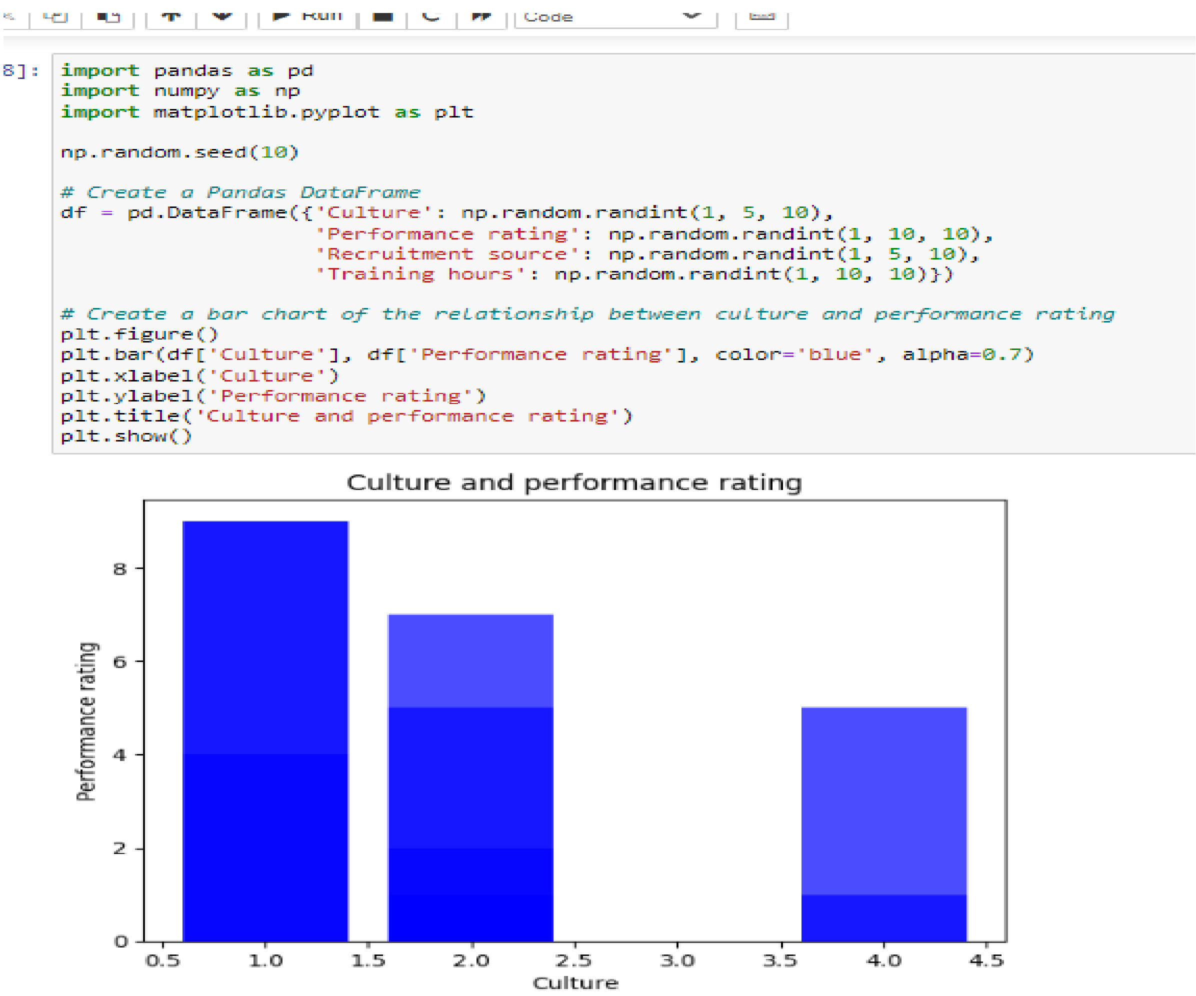
Out[16]:

	HCRI	Training Score	Performance Score
HCRI	1.000000	-0.044348	0.051114
Training Score	-0.044348	1.000000	0.029658
Performance Score	0.051114	0.029658	1.000000



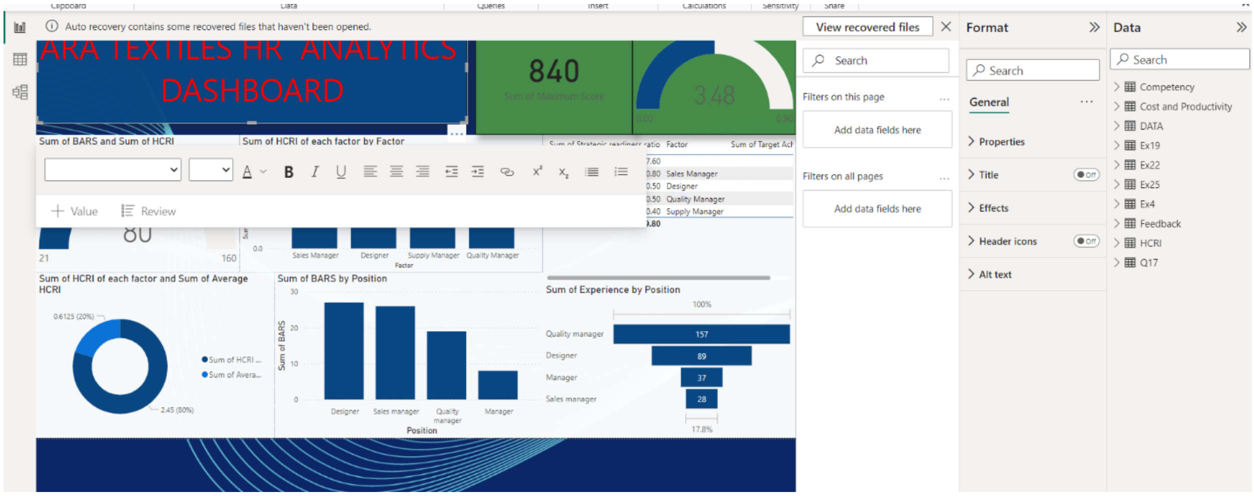
```
In [17]: import seaborn as sns
         import matplotlib.pyplot as plt

         # Create a heatmap of the correlation matrix
         plt.figure(figsize=(22, 10))
         sns.heatmap(cor_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
         plt.title("Correlation Heatmap")
         plt.show()
```



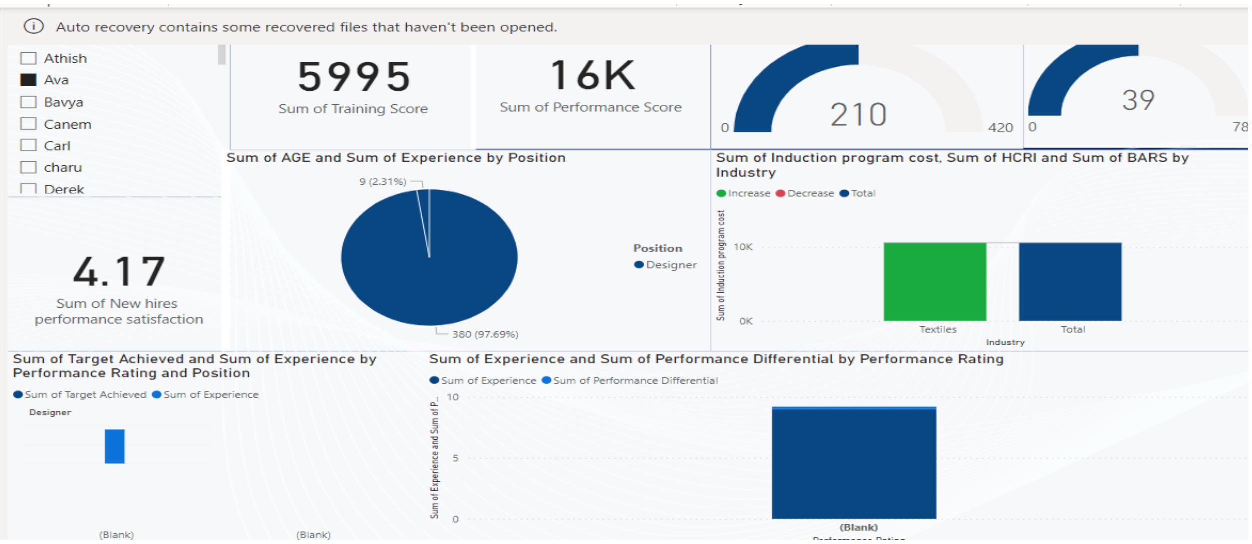


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(10)

# Create a Pandas DataFrame
df = pd.DataFrame({'Culture': np.random.randint(1, 5, 10),
                   'Performance rating': np.random.randint(1, 10, 10),
                   'Recruitment source': np.random.randint(1, 5, 10),
                   'Training hours': np.random.randint(1, 10, 10)})

# Create a bar chart of the relationship between culture and performance rating
plt.figure()
plt.bar(df['Culture'], df['Performance rating'], color='blue', alpha=0.7)
plt.xlabel('Culture')
plt.ylabel('Performance rating')
plt.title('Culture and performance rating')
plt.show()
```

DASHBOARDS:

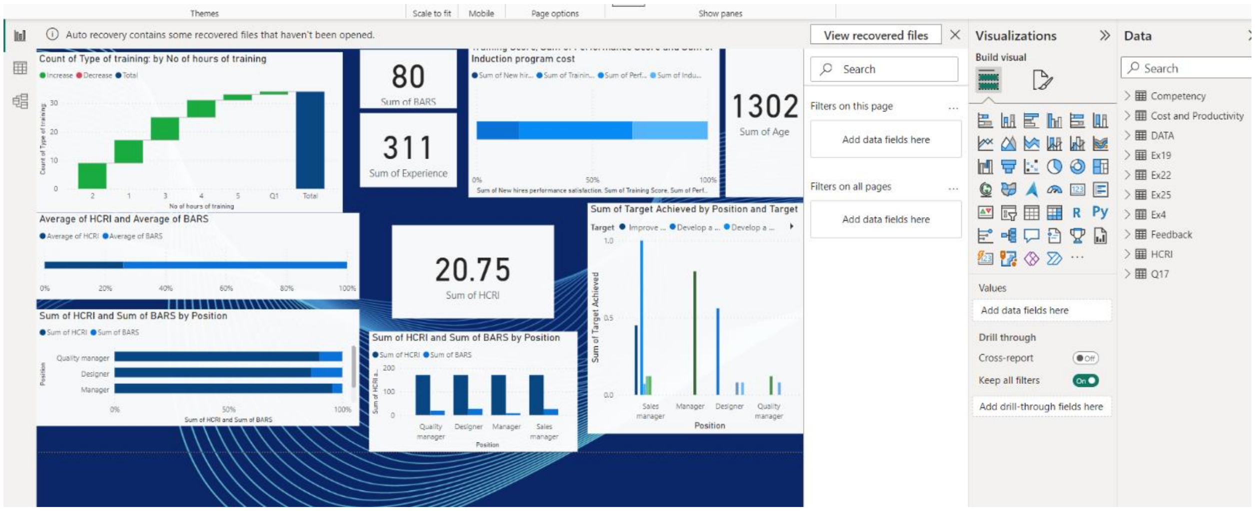


Auto recovery contains some recovered files that haven't been opened.



Auto recovery contains some recovered files that haven't been opened.

Athish
Ava
Bavya
Canem
Carl
charu
Derek
5995
Sum of Training Score
16K
Sum of Performance Score
0
210
420
0
39
78
4.17
Sum of New hires performance satisfaction
Sum of AGE and Sum of Experience by Position
9 (2.31%)
380 (97.69%)
Position
Designer
Sum of Induction program cost, Sum of HCRI and Sum of BARS by Industry
Increase
Decrease
Total
Sum of Induction program cost
10K
0K
Textiles
Total
Industry
Sum of Target Achieved and Sum of Experience by Performance Rating and Position
Sum of Target Achieved
Sum of Experience
Designer
(Blank)
(Blank)
Sum of Experience and Sum of Performance Differential by Performance Rating
Sum of Experience
Sum of Performance Differential
Sum of Experience and Sum of P...
10
5
0
(Blank)
Performance Rating



Auto recovery contains some recovered files that haven't been opened.