Engagement & Performance: What factors predict high performance among employees?

Which factors most strongly predict an employee's performance rating?

In [65]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('Extended_Employee_Performance_and_Productivity_Data.csv')
print(df.shape)
df.head()
```

```
(100000, 20)
```

Out[65]:

	Employee_ID	Department	Gender	Age	Job_Title	Hire_Date	Years_At_Company	Education_
0	1	IT	Male	55	Specialist	2022-01-19 08:03:05.556036	2	High
1	2	Finance	Male	29	Developer	2024-04-18 08:03:05.556036	0	High
2	3	Finance	Male	55	Specialist	2015-10-26 08:03:05.556036	8	High
3	4	Customer Support	Female	48	Analyst	2016-10-22 08:03:05.556036	7	Ba
4	5	Engineering	Female	36	Analyst	2021-07-23 08:03:05.556036	3	Ba

In [67]:
```
df.dtypes
```

Out[67]:

```
Employee_ID                    int64
Department                    object
Gender                        object
Age                            int64
Job_Title                     object
Hire_Date                     object
Years_At_Company               int64
Education_Level               object
Performance_Score              int64
Monthly_Salary               float64
Work_Hours_Per_Week            int64
Projects_Handled               int64
Overtime_Hours                 int64
Sick_Days                      int64
Remote_Work_Frequency          int64
Team_Size                      int64
Training_Hours                 int64
Promotions                     int64
Employee_Satisfaction_Score  float64
Resigned                        bool
dtype: object
```

In [68]:
```
for col in df.columns:
    print(col, ":", df[col].nunique())
```

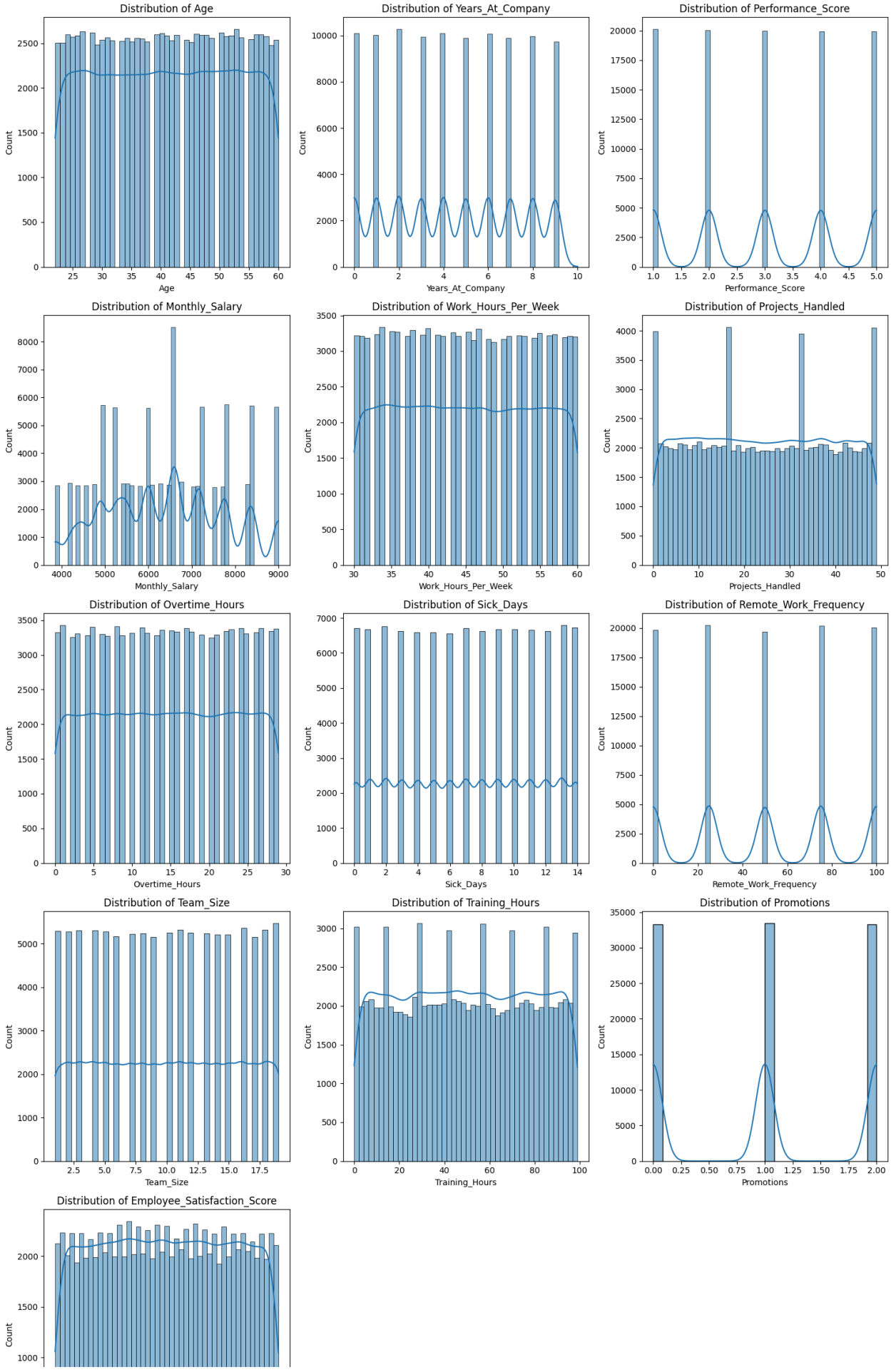
```
Employee_ID : 100000
Department : 9
Gender : 3
Age : 39
Job_Title : 7
Hire_Date : 3650
Years_At_Company : 11
Education_Level : 4
Performance_Score : 5
Monthly_Salary : 28
Work_Hours_Per_Week : 31
Projects_Handled : 50
Overtime_Hours : 30
Sick_Days : 15
Remote_Work_Frequency : 5
Team_Size : 19
Training_Hours : 100
Promotions : 3
Employee_Satisfaction_Score : 401
Resigned : 2
```

In [74]:
```python
if 'Hire_Date' in df.columns:
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], errors='coerce')

print("\nMissing values count:\n", df.isnull().sum())
```

```
Missing values count:
 Department                    0
Gender                         0
Age                            0
Job_Title                      0
Hire_Date                      0
Years_At_Company               0
Education_Level                0
Performance_Score              0
Monthly_Salary                 0
Work_Hours_Per_Week            0
Projects_Handled               0
Overtime_Hours                 0
Sick_Days                      0
Remote_Work_Frequency          0
Team_Size                      0
Training_Hours                 0
Promotions                     0
Employee_Satisfaction_Score    0
Resigned                       0
dtype: int64
```

In [71]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

num_cols = len(numeric_cols)
num_rows = (num_cols // 3) + (num_cols % 3 > 0)

fig, axes = plt.subplots(num_rows, 3, figsize=(15, 5 * num_rows))

axes = axes.flatten()

for i, col in enumerate(numeric_cols):
    sns.histplot(data=df, x=col, kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}')

for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])
```

```
plt.tight_layout()
plt.show()
```


Distribution of Age
Count
Age
Distribution of Years_At_Company
Count
Years_At_Company
Distribution of Performance_Score
Count
Performance_Score
Distribution of Monthly_Salary
Count
Monthly_Salary
Distribution of Work_Hours_Per_Week
Count
Work_Hours_Per_Week
Distribution of Projects_Handled
Count
Projects_Handled
Distribution of Overtime_Hours
Count
Overtime_Hours
Distribution of Sick_Days
Count
Sick_Days
Distribution of Remote_Work_Frequency
Count
Remote_Work_Frequency
Distribution of Team_Size
Count
Team_Size
Distribution of Training_Hours
Count
Training_Hours
Distribution of Promotions
Count
Promotions
Distribution of Employee_Satisfaction_Score
Count

In [72]:
```
print("\nValue Counts for Performance_Score:\n", df['Performance_Score'].value_counts())
```

```
Value Counts for Performance_Score:
 Performance_Score
1    20120
2    20013
3    19999
4    19940
5    19928
Name: count, dtype: int64
```

In [75]:
```
y = df['Performance_Score']

X = df.drop(['Performance_Score', 'Employee_ID'], axis=1, errors='ignore')
```

In [76]:
```
cat_cols = X.select_dtypes(include=['object', 'category']).columns
print("Categorical columns:", cat_cols)

X = pd.get_dummies(X, columns=cat_cols, drop_first=True)
```

```
Categorical columns: Index(['Department', 'Gender', 'Job_Title', 'Education_Level'], dtype
='object')
```

In [78]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.20,
    random_state=42,
    stratify=y
)

print("Train shape:", X_train.shape, "Test shape:", X_test.shape)
```

```
Train shape: (80000, 33) Test shape: (20000, 33)
```

Random Forest

In [81]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

X_train = X_train.select_dtypes(include=['number'])
X_test = X_test.select_dtypes(include=['number'])

rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=None,
    random_state=42
)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.78225
Classification Report:
              precision    recall  f1-score   support

           1       0.57      0.90      0.70      4024
           2       0.76      0.67      0.72      4003
           3       0.86      0.80      0.83      4000
           4       0.94      0.78      0.86      3988
           5       0.96      0.75      0.85      3985

    accuracy                           0.78     20000
   macro avg       0.82      0.78      0.79     20000
weighted avg       0.82      0.78      0.79     20000

Confusion Matrix:
 [[3636  224  106   39   19]
 [1082 2700  113   49   59]
 [ 559  183 3194   48   16]
 [ 575  139  140 3119   15]
 [ 507  287  149   46 2996]]
```

With Hyperparameter Tuning

In [82]:

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    scoring='accuracy',
    cv=5,
    n_jobs=-1,
    verbose=1
)

grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_
print("Best parameters:", grid_search.best_params_)

y_pred_gs = best_rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred_gs))
print("Classification Report:\n", classification_report(y_test, y_pred_gs))
```

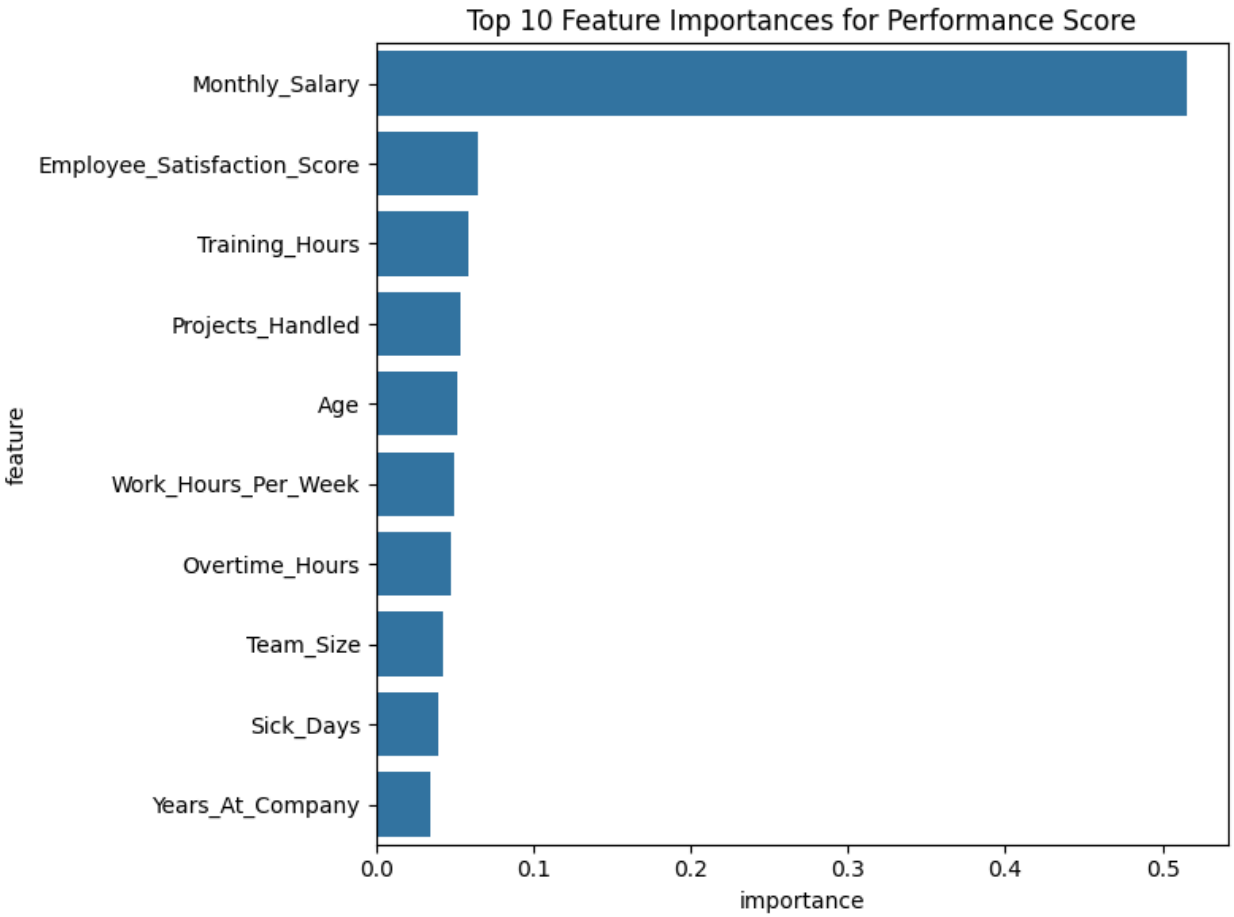
```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
/Users/stevenchen/Library/Python/3.9/lib/python/site-packages/joblib/externals/loky/process_
executor.py:752: UserWarning: A worker stopped while some jobs were given to the executor. T
his can be caused by a too short worker timeout or by a memory leak.
  warnings.warn(
```

```
Best parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 200}
Accuracy: 0.80495
Classification Report:
              precision    recall  f1-score   support

           1       0.58      0.95      0.72      4024
           2       0.80      0.69      0.74      4003
           3       0.90      0.83      0.86      4000
           4       0.98      0.80      0.88      3988
           5       0.99      0.75      0.85      3985

    accuracy                           0.80     20000
   macro avg       0.85      0.80      0.81     20000
weighted avg       0.85      0.80      0.81     20000
```

In [95]:

```python
importances = best_rf.feature_importances_
feature_names = X_train.columns


feat_imp_df = pd.DataFrame({
    'feature': feature_names,
    'importance': importances
}).sort_values('importance', ascending=False)

print("\nTop 10 Factors Impacting Performance_Score:")
print(feat_imp_df.head(10))

plt.figure(figsize=(8,6))
sns.barplot(data=feat_imp_df.head(10), x='importance', y='feature')
plt.title('Top 10 Feature Importances for Performance Score')
plt.tight_layout()
plt.show()
```

```
Top 10 Factors Impacting Performance_Score:
                       feature  importance
2                Monthly_Salary    0.515963
11  Employee_Satisfaction_Score    0.064552
9                Training_Hours    0.059181
4              Projects_Handled    0.053640
0                          Age    0.051847
3           Work_Hours_Per_Week    0.049759
5                Overtime_Hours    0.048183
8                     Team_Size    0.042804
6                     Sick_Days    0.039406
1              Years_At_Company    0.034479
```



Key Factors Affecting Performance Score

Monthly Salary stands out significantly among all predictors, suggesting that compensation is closely tied to performance. Employees drawing higher salaries may hold more advanced or senior positions, which naturally correlates with higher performance ratings. This highlights a pay-for-performance structure or an organizational culture where monetary rewards align with perceived contributions.

Beyond salary, several additional factors also influence performance. Employee Satisfaction and Training Hours indicate that engagement and development are meaningful drivers of success—invested, well-trained employees typically demonstrate stronger performance. Projects Handled, Work Hours, and Overtime Hours further suggest that employees who take on heavier workloads or more complex responsibilities are perceived as higher performers, though this balance must be managed to avoid burnout. Age, along with Years at the Company, appears to function as a proxy for accumulated experience, but it should be interpreted carefully to avoid assumptions about an employee's capabilities based solely on tenure or age.

Finally, Team Size and Sick Days point to the importance of interpersonal dynamics and attendance. Leading or participating in larger teams can present both challenges and opportunities that positively impact performance appraisals, while lower absence rates often promote continuity and reliability within a role. Overall, while monetary compensation is a major influence, engagement, training, workload distribution, and team dynamics collectively shape an employee's performance score.