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

```
df = pd.read_csv("employee_productivity.csv")
```

```
df.head()
```

| | Employee_ID | Department | Role | Experience_Years | Working_Hours | Attendance_Rate | Tasks_Completed | Performance_Score |
|---|---|---|---|---|---|---|---|---|
| **0** | E001 | IT | Software Engineer | 2.0 | 8.0 | 95 | 120 | 8 |
| **1** | E002 | IT | Software Engineer | 5.0 | 9.0 | 92 | 140 | 9 |
| **2** | E003 | HR | HR Executive | 3.0 | 7.5 | 90 | 100 | 7 |

Next steps: Generate code with df | New interactive sheet

```
df.shape
```

```
(20, 10)
```

```
df.columns
```

```
Index(['Employee_ID', 'Department', 'Role', 'Experience_Years',
       'Working_Hours', 'Attendance_Rate', 'Tasks_Completed',
       'Performance_Score', 'Overtime_Hours', 'Salary'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Employee_ID        20 non-null     object 
 1   Department         20 non-null     object 
 2   Role               20 non-null     object 
 3   Experience_Years   20 non-null     float64
 4   Working_Hours      20 non-null     float64
 5   Attendance_Rate    20 non-null     int64  
 6   Tasks_Completed    20 non-null     int64  
 7   Performance_Score  20 non-null     int64  
 8   Overtime_Hours     20 non-null     int64  
 9   Salary             20 non-null     int64  
dtypes: float64(2), int64(5), object(3)
memory usage: 1.7+ KB
```

```
df.isnull().sum()
```

| | 0 |
|---|---|
| Employee_ID | 0 |

```
df.describe()
```

| | Experience_Years | Working_Hours | Attendance_Rate | Tasks_Completed | Performance_Score | Overtime_Hours | Salary |
|---|---|---|---|---|---|---|---|
| count | 20.000000 | 20.000000 | 20.000000 | 20.000000 | 20.000000 | 20.000000 | 20.000000 |
| mean | 4.020000 | 7.800000 | 91.000000 | 120.800000 | 7.550000 | 10.650000 | 42950.000000 |
| std | 2.892568 | 0.951453 | 3.509386 | 19.033488 | 1.050063 | 6.482974 | 20085.278715 |
| min | 0.500000 | 6.000000 | 85.000000 | 90.000000 | 6.000000 | 2.000000 | 15000.000000 |
| 25% | 1.800000 | 7.375000 | 88.750000 | 103.750000 | 7.000000 | 5.750000 | 29500.000000 |
| 50% | 3.500000 | 8.000000 | 90.500000 | 122.500000 | 8.000000 | 10.000000 | 41500.000000 |
| 75% | 6.000000 | 8.500000 | 94.000000 | 132.750000 | 8.000000 | 14.250000 | 56250.000000 |
| max | 10.000000 | 9.000000 | 97.000000 | 155.000000 | 9.000000 | 25.000000 | 85000.000000 |

```
df
```

| | Employee_ID | Department | Role | Experience_Years | Working_Hours | Attendance_Rate | Tasks_Completed | Performance_Score |
|---|---|---|---|---|---|---|---|---|
| 0 | E001 | IT | Software Engineer | 2.0 | 8.0 | 95 | 120 | 8 |
| 1 | E002 | IT | Software Engineer | 5.0 | 9.0 | 92 | 140 | 9 |
| 2 | E003 | HR | HR Executive | 3.0 | 7.5 | 90 | 100 | 7 |
| 3 | E004 | HR | HR Manager | 8.0 | 8.0 | 96 | 130 | 9 |
| 4 | E005 | Sales | Sales Executive | 1.0 | 8.5 | 85 | 110 | 6 |
| 5 | E006 | Sales | Sales Manager | 6.0 | 9.0 | 88 | 150 | 8 |
| 6 | E007 | Finance | Accountant | 4.0 | 8.0 | 94 | 125 | 8 |
| 7 | E008 | Finance | Senior Accountant | 10.0 | 8.5 | 97 | 145 | 9 |
| 8 | E009 | Operations | Operations Executive | 2.0 | 7.0 | 89 | 105 | 7 |
| 9 | E010 | Operations | Operations Manager | 7.0 | 8.5 | 93 | 135 | 8 |
| 10 | E011 | IT | Data Analyst | 3.0 | 8.0 | 91 | 128 | 8 |
| 11 | E012 | IT | Team Lead | 9.0 | 9.0 | 95 | 155 | 9 |
| 12 | E013 | Sales | Sales Executive | 2.0 | 8.0 | 87 | 115 | 7 |
| 13 | E014 | HR | Recruiter | 4.0 | 7.5 | 92 | 118 | 8 |

Next steps: Generate code with df | New interactive sheet

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
df['Tasks_norm'] = df['Tasks_Completed'] / df['Tasks_Completed'].max()
df['Attendance_norm'] = df['Attendance_Rate'] / 100
df['Performance_norm'] = df['Performance_Score'] / 10
```

```
df['Productivity_Score'] = (
    df['Tasks_norm'] * 0.4 +
    df['Attendance_norm'] * 0.3 +
    df['Performance_norm'] * 0.3
) * 100
```

```
df[['Employee_ID', 'Department', 'Role', 'Productivity_Score']].head()
```

| | Employee_ID | Department | Role | Productivity_Score |
|---|---|---|---|---|
| 0 | E001 | IT | Software Engineer | 83.467742 |
| 1 | E002 | IT | Software Engineer | 90.729032 |
| 2 | E003 | HR | HR Executive | 73.806452 |
| 3 | E004 | HR | HR Manager | 89.348387 |
| 4 | E005 | Sales | Sales Executive | 71.887097 |

```
def productivity_level(score):
    if score >= 75:
        return 'High'
    elif score >= 50:
        return 'Medium'
    else:
        return 'Low'

df['Productivity_Level'] = df['Productivity_Score'].apply(productivity_level)
```

```
df['Productivity_Level'].value_counts()
```

| Productivity_Level | count |
|---|---|
| High | 13 |
| Medium | 7 |

**dtype:** int64

```
df.groupby('Department')['Productivity_Score'].mean().sort_values(ascending=False)
```

| Department | Productivity_Score |
|---|---|
| Finance | 84.512097 |
| IT | 84.330968 |
| Operations | 80.597849 |
| HR | 78.799194 |
| Sales | 76.447581 |

**dtype:** float64

```
df[['Experience_Years', 'Productivity_Score']].corr()
```

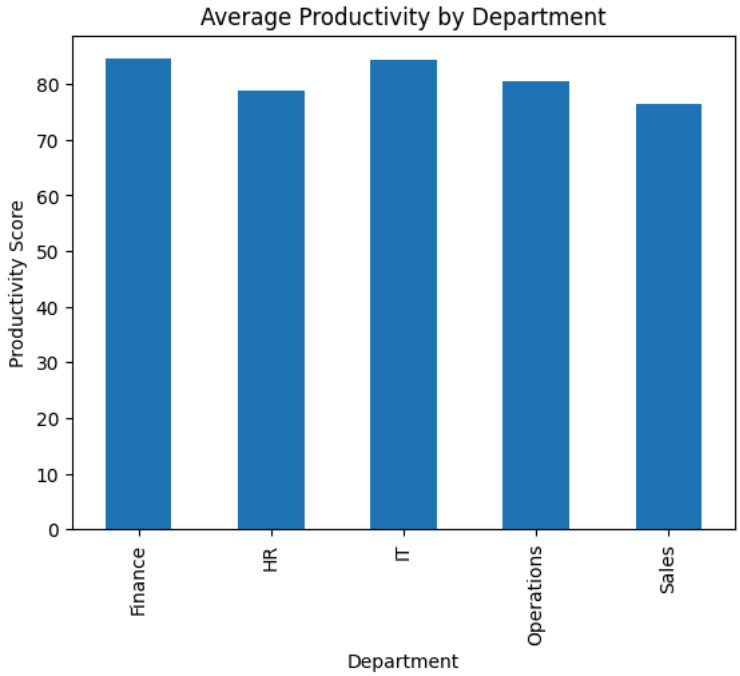
| | Experience_Years | Productivity_Score |
|---|---|---|
| Experience_Years | 1.000000 | 0.885792 |
| Productivity_Score | 0.885792 | 1.000000 |

```
df[['Working_Hours', 'Productivity_Score']].corr()
```

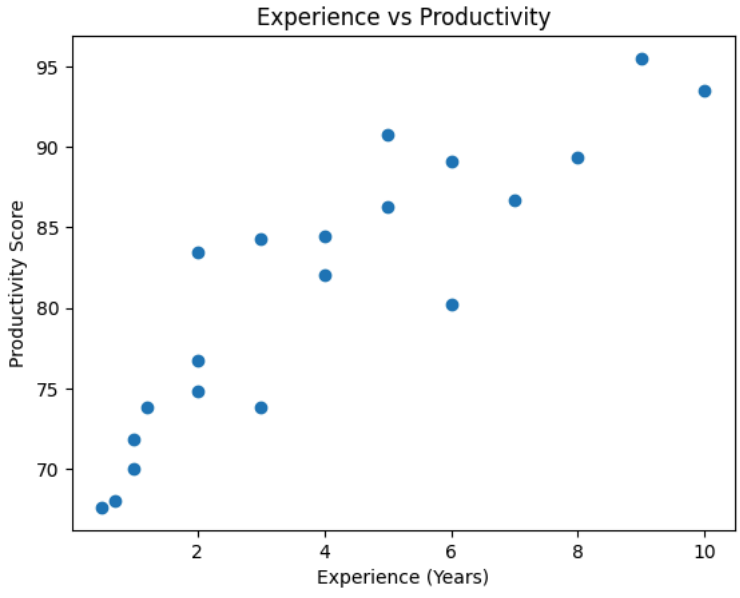
| | Working_Hours | Productivity_Score |
|---|---|---|
| **Working_Hours** | 1.000000 | 0.827179 |
| **Productivity_Score** | 0.827179 | 1.000000 |

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

```
df.groupby('Department')['Productivity_Score'].mean().plot(kind='bar')
plt.title('Average Productivity by Department')
plt.ylabel('Productivity Score')
plt.xlabel('Department')
plt.show()
```

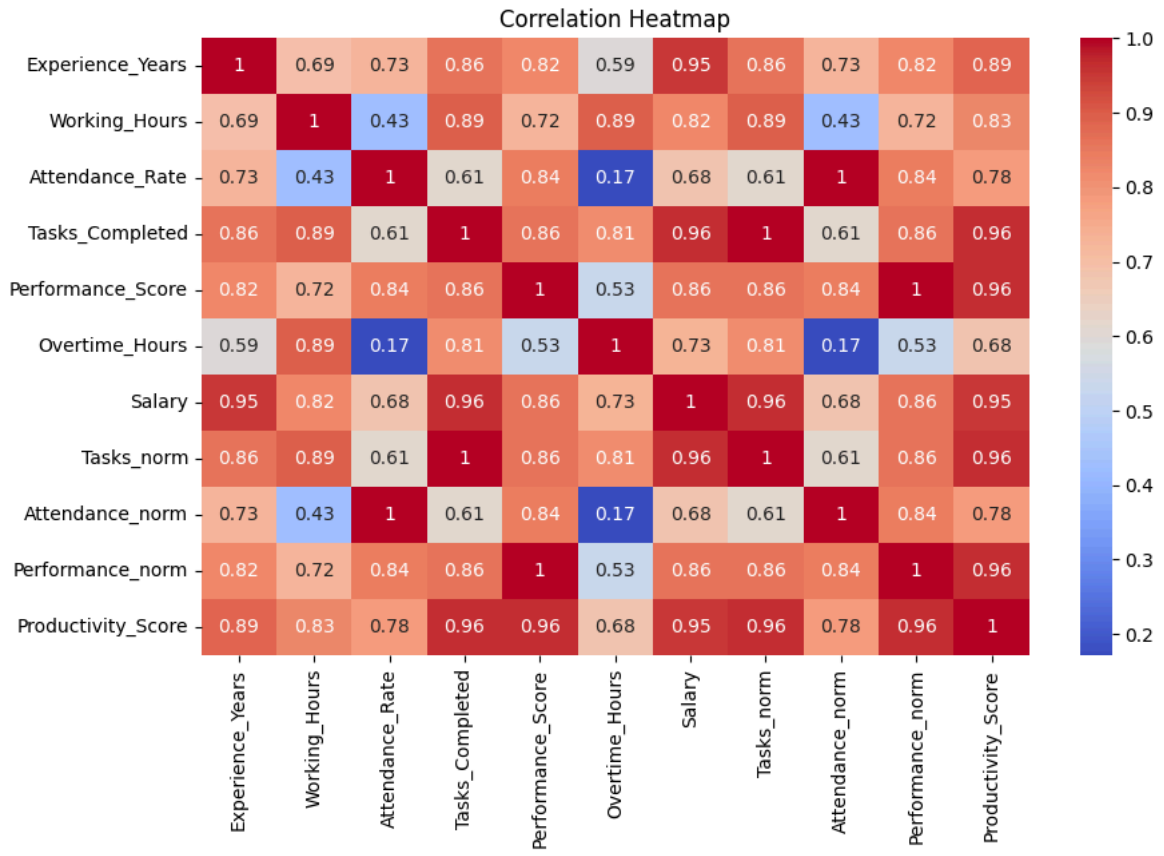



```
plt.scatter(df['Experience_Years'], df['Productivity_Score'])
plt.xlabel('Experience (Years)')
plt.ylabel('Productivity Score')
plt.title('Experience vs Productivity')
plt.show()
```




```
plt.figure(figsize=(10,6))
sns.heatmap(df.select_dtypes(include=np.number).corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
```

```
plt.title('Correlation Heatmap')
plt.show()
```




**Insights:**

High attendance strongly improves productivity

Excess overtime negatively impacts performance

Moderate working hours give best results

Experience increases productivity up to a limit

**Recommendations:**

Encourage work-life balance

Reduce excessive overtime

Introduce performance-based incentives

Conduct department-specific training

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.