```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import IsolationForest
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Step 1: Load the CSV file
df = pd.read_csv('/content/employee_data.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   EmployeeID         200 non-null    object
 1   Age                180 non-null    float64
 2   Department         180 non-null    object
 3   Salary             180 non-null    float64
 4   JoiningDate        200 non-null    object
 5   ExperienceYears    180 non-null    float64
 6   PerformanceRating  200 non-null    float64
 7   Gender             200 non-null    object
 8   OvertimeHours      180 non-null    float64
 9   WorkFromHome       180 non-null    object
 10  LeftCompany        200 non-null    object
 11  YearsInCompany     200 non-null    float64
 12  OvertimeEffect     180 non-null    float64
dtypes: float64(7), object(6)
memory usage: 20.4+ KB
```

```
df.describe()
```

	Age	Salary	ExperienceYears	PerformanceRating	OvertimeHours	YearsInCompa
count	1.800000e+02	1.800000e+02	1.800000e+02	2.000000e+02	1.800000e+02	2.000000e+
mean	-1.973730e-16	4.440892e-17	-1.233581e-16	-5.773160e-17	5.921189e-17	1.643130e-
std	1.002789e+00	1.002789e+00	1.002789e+00	1.002509e+00	1.002789e+00	1.002509e+
min	-1.638947e+00	-6.523035e-01	-1.696490e+00	-1.452387e+00	-1.318474e+00	-1.894146e+
25%	-9.254995e-01	-3.848327e-01	-7.787633e-01	-7.456296e-01	-5.898994e-01	-8.112612e-
50%	5.549033e-02	-1.442968e-01	5.359333e-02	-3.887168e-02	-5.138742e-02	9.114280e-
75%	7.912327e-01	1.553473e-01	7.792376e-01	6.678862e-01	5.821561e-01	9.935468e-
max	1.660746e+00	1.050523e+01	3.681814e+00	1.374644e+00	7.551135e+00	1.715470e+

```
# Step 3: Handle Missing Values
# Impute missing numerical values using the median (more robust to outliers)
imputer_num = SimpleImputer(strategy='median')
df[['Age', 'Salary', 'ExperienceYears', 'PerformanceRating', 'OvertimeHours']] = imputer_num.fit_trans
    df[['Age', 'Salary', 'ExperienceYears', 'PerformanceRating', 'OvertimeHours']]
)


# Impute missing categorical values using the most frequent value
imputer_cat = SimpleImputer(strategy='most_frequent')
df[['Department', 'Gender', 'WorkFromHome']] = imputer_cat.fit_transform(df[['Department', 'Gender', '


# Step 4: Handle Outliers
# Use Isolation Forest to detect and remove outliers based on numerical features
iso_forest = IsolationForest(contamination=0.02, random_state=42)
outliers = iso_forest.fit_predict(df[['Age', 'Salary', 'ExperienceYears', 'PerformanceRating', 'Overti
df=df[outliers == 1]  # Keep only inliers
df
```

	EmployeeID	Age	Department	Salary	JoiningDate	ExperienceYears	PerformanceRatin
0	E0001	0.858118	HR	-0.144297	2015-08-23 11:12:07.839953	-0.415941	0.66788
1	E0002	-0.390414	HR	-0.193042	2007-01-22 11:12:07.839983	0.181648	-1.45238
2	E0003	-1.014680	IT	0.101163	2007-12-30 11:12:07.839986	0.053593	-1.45238
3	E0004	0.144671	Finance	-0.144297	2015-06-02 11:12:07.839988	-0.757421	1.37464
4	E0005	-0.033691	HR	3.628184	2019-06-18 11:12:07.839990	-0.928161	0.66788
...	...	...	...	...	...	...	.
194	E0195	-1.460585	HR	-0.455321	2013-06-22 11:12:07.840417	-0.928161	-0.74563
195	E0196	-0.033691	Admin	-0.048962	2015-09-25 11:12:07.840419	1.547567	-1.45238
196	E0197	0.055490	Sales	-0.213367	2013-04-28 11:12:07.840421	-0.330571	-0.03887
197	E0198	1.125661	Sales	-0.047681	2018-12-02 11:12:07.840423	1.547567	1.37464
198	E0199	-1.103861	Admin	-0.046539	2009-06-01 11:12:07.840425	3.681814	-1.45238

196 rows × 13 columns

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

```
df['EmployeeID'] = df['EmployeeID'].str.replace('E', '').str[-3:]
df
```

	EmployeeID	Age	Department	Salary	JoiningDate	ExperienceYears	PerformanceRatin
0	001	0.858118	HR	-0.144297	2015-08-23 11:12:07.839953	-0.415941	0.66788
1	002	-0.390414	HR	-0.193042	2007-01-22 11:12:07.839983	0.181648	-1.45238
2	003	-1.014680	IT	0.101163	2007-12-30 11:12:07.839986	0.053593	-1.45238
3	004	0.144671	Finance	-0.144297	2015-06-02 11:12:07.839988	-0.757421	1.37464
4	005	-0.033691	HR	3.628184	2019-06-18 11:12:07.839990	-0.928161	0.66788
...	...	...	...	...	...	...	.
194	195	-1.460585	HR	-0.455321	2013-06-22 11:12:07.840417	-0.928161	-0.74563
195	196	-0.033691	Admin	-0.048962	2015-09-25 11:12:07.840419	1.547567	-1.45238
196	197	0.055490	Sales	-0.213367	2013-04-28 11:12:07.840421	-0.330571	-0.03887
197	198	1.125661	Sales	-0.047681	2018-12-02 11:12:07.840423	1.547567	1.37464
198	199	-1.103861	Admin	-0.046539	2009-06-01 11:12:07.840425	3.681814	-1.45238

196 rows × 13 columns

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

```
# Step 5: Encoding Categorical Variables
# Label encode binary categorical variables
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])  # Male=1, Female=0
df['WorkFromHome'] = label_encoder.fit_transform(df['WorkFromHome'])  # Yes=1, No=0
df['LeftCompany'] = label_encoder.fit_transform(df['LeftCompany'])  # No=0, Yes=1


# One-hot encode categorical variables with more than two categories (e.g., 'Department')
df = pd.get_dummies(df, columns=['Department'], drop_first=True)


# Step 6: Handling Date Variables
# Convert 'JoiningDate' to datetime and extract useful features like year, month, and day
df['JoiningDate'] = pd.to_datetime(df['JoiningDate'])
df['JoiningYear'] = df['JoiningDate'].dt.year
df['JoiningMonth'] = df['JoiningDate'].dt.month
df['JoiningDay'] = df['JoiningDate'].dt.day
df = df.drop('JoiningDate', axis=1)


# Step 7: Feature Engineering
# Create a new feature: 'YearsInCompany' by subtracting experience from total years worked
df['YearsInCompany'] = df['JoiningYear'].max() - df['JoiningYear']
```

```
# Create an interaction feature: 'OvertimeEffect' (OvertimeHours * PerformanceRating)
df['OvertimeEffect'] = df['OvertimeHours'] * df['PerformanceRating']


# Step 8: Feature Scaling
# Standardize numerical features using StandardScaler
scaler = StandardScaler()
df[['Age', 'Salary', 'ExperienceYears', 'PerformanceRating', 'OvertimeHours', 'YearsInCompany', 'Overt
    df[['Age', 'Salary', 'ExperienceYears', 'PerformanceRating', 'OvertimeHours', 'YearsInCompany', 'C
)



sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
#plt.title('Correlation Matrix')
#plt.show()



# Step 9: Handling Imbalanced Data
# Use SMOTE (Synthetic Minority Over-sampling Technique) to balance the target class
X = df.drop(['EmployeeID', 'LeftCompany'], axis=1)  # Features
y = df['LeftCompany']  # Target variable
print(X)
```

```
          Age    Salary  ExperienceYears  PerformanceRating  Gender  \
0    0.896217 -0.109543        -0.441970           0.668063       1
1   -0.424524 -0.221013         0.184458          -1.443878       1
2   -1.084895  0.451773         0.050224          -1.443878       0
3    0.141508 -0.109543        -0.799929           1.372043       1
4   -0.047169  8.517342        -0.978908           0.668063       1
..        ...       ...              ...                ...     ...
194 -1.556588 -0.820791        -0.978908          -0.739897       0
195 -0.047169  0.108468         1.616294          -1.443878       1
196  0.047169 -0.267492        -0.352480          -0.035917       1
197  1.179234  0.111398         1.616294           1.372043       1
198 -1.179234  0.114008         3.853537          -1.443878       0

     OvertimeHours  WorkFromHome  YearsInCompany  OvertimeEffect  \
0        -0.138968             0       -1.439809       -0.297713
1        -1.673485             1        1.101030        2.425098
2        -1.481670             1        0.592862        2.152618
3         0.244661             0        1.439809        0.037086
4         1.779178             1       -1.609198        0.955297
..             ...           ...             ...             ...
194      -1.481670             1        1.439809        1.043472
195       0.436476             0       -0.931641       -0.572182
196       0.628291             0       -1.439809       -0.145894
197       1.395549             1        0.254084        1.584454
198      -1.289855             0        0.931641        1.880138

     Department_Finance  Department_HR  Department_IT  Department_Sales  \
0                 False           True          False             False
1                 False           True          False             False
2                 False          False           True             False
3                  True          False          False             False
4                 False           True          False             False
..                  ...            ...            ...               ...
194               False           True          False             False
195               False          False          False             False
196               False          False          False              True
197               False          False          False              True
198               False          False          False             False

     JoiningYear  JoiningMonth  JoiningDay
0           2023             6          25
1           2008             9          27
```

```
2        2011      8     31
3        2006      6     15
4        2024      3      7
..        ...    ...    ...
194      2006     12      1
195      2020      2     25
196      2023      6     29
197      2013     11      5
198      2009     12     18

[196 rows x 16 columns]
```

```
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
# Step 10: Splitting the Dataset into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_st
```

```
# Step 11: Correlation Analysis (optional for feature selection)
plt.figure(figsize=(12, 8))
df.corr()
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```