HR Employee Attrition

Load and Explore Data

In [23]:
```
import pandas as pd
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report


# Load your dataset
df = pd.read_csv("HR-Employee-Attrition.csv")
```

In [24]:
```
# Check for duplicates
duplicate_rows = df.duplicated().sum()

# Check for missing values
missing_values = df.isnull().sum()

# Get summary statistics
summary_statistics = df.describe()

# Get unique values per column
unique_values = df.nunique()

# Data types overview
data_types = df.dtypes

# Results
data_overview = {
    "duplicate_rows": duplicate_rows,
    "missing_values": missing_values[missing_values > 0].to_dict(),  # Only show co
    "summary_statistics": summary_statistics.to_dict(),
    "unique_values": unique_values.to_dict(),
    "data_types": data_types.to_dict(),
}

data_overview
```

Out[24]:
```
{'duplicate_rows': 0,
 'missing_values': {},
 'summary_statistics': {'Age': {'count': 1470.0,
   'mean': 36.923809523809524,
   'std': 9.135373489136734,
   'min': 18.0,
   '25%': 30.0,
   '50%': 36.0,
   '75%': 43.0,
   'max': 60.0},
  'DailyRate': {'count': 1470.0,
   'mean': 802.4857142857143,
   'std': 403.50909994352827,
   'min': 102.0,
   '25%': 465.0,
   '50%': 802.0,
   '75%': 1157.0,
   'max': 1499.0},
  'DistanceFromHome': {'count': 1470.0,
   'mean': 9.19251700680272,
   'std': 8.106864435666084,
   'min': 1.0,
   '25%': 2.0,
   '50%': 7.0,
   '75%': 14.0,
   'max': 29.0},
  'Education': {'count': 1470.0,
   'mean': 2.912925170068027,
   'std': 1.0241649445978729,
   'min': 1.0,
   '25%': 2.0,
   '50%': 3.0,
   '75%': 4.0,
   'max': 5.0},
  'EmployeeCount': {'count': 1470.0,
   'mean': 1.0,
   'std': 0.0,
   'min': 1.0,
   '25%': 1.0,
   '50%': 1.0,
   '75%': 1.0,
   'max': 1.0},
  'EmployeeNumber': {'count': 1470.0,
   'mean': 1024.865306122449,
   'std': 602.024334847475,
   'min': 1.0,
   '25%': 491.25,
   '50%': 1020.5,
   '75%': 1555.75,
   'max': 2068.0},
  'EnvironmentSatisfaction': {'count': 1470.0,
   'mean': 2.721768707482993,
   'std': 1.0930822146349994,
   'min': 1.0,
   '25%': 2.0,
   '50%': 3.0,
```

```
 '75%': 4.0,
 'max': 4.0},
'HourlyRate': {'count': 1470.0,
 'mean': 65.89115646258503,
 'std': 20.329427593996158,
 'min': 30.0,
 '25%': 48.0,
 '50%': 66.0,
 '75%': 83.75,
 'max': 100.0},
'JobInvolvement': {'count': 1470.0,
 'mean': 2.7299319727891156,
 'std': 0.7115611429632283,
 'min': 1.0,
 '25%': 2.0,
 '50%': 3.0,
 '75%': 3.0,
 'max': 4.0},
'JobLevel': {'count': 1470.0,
 'mean': 2.0639455782312925,
 'std': 1.1069398989351114,
 'min': 1.0,
 '25%': 1.0,
 '50%': 2.0,
 '75%': 3.0,
 'max': 5.0},
'JobSatisfaction': {'count': 1470.0,
 'mean': 2.7285714285714286,
 'std': 1.1028461230547149,
 'min': 1.0,
 '25%': 2.0,
 '50%': 3.0,
 '75%': 4.0,
 'max': 4.0},
'MonthlyIncome': {'count': 1470.0,
 'mean': 6502.931292517007,
 'std': 4707.956783097995,
 'min': 1009.0,
 '25%': 2911.0,
 '50%': 4919.0,
 '75%': 8379.0,
 'max': 19999.0},
'MonthlyRate': {'count': 1470.0,
 'mean': 14313.103401360544,
 'std': 7117.786044059973,
 'min': 2094.0,
 '25%': 8047.0,
 '50%': 14235.5,
 '75%': 20461.5,
 'max': 26999.0},
'NumCompaniesWorked': {'count': 1470.0,
 'mean': 2.6931972789115646,
 'std': 2.498009006070751,
 'min': 0.0,
 '25%': 1.0,
 '50%': 2.0,
```

```
 '75%': 4.0,
 'max': 9.0},
'PercentSalaryHike': {'count': 1470.0,
 'mean': 15.209523809523809,
 'std': 3.659937716539636,
 'min': 11.0,
 '25%': 12.0,
 '50%': 14.0,
 '75%': 18.0,
 'max': 25.0},
'PerformanceRating': {'count': 1470.0,
 'mean': 3.1537414965986397,
 'std': 0.36082352460434486,
 'min': 3.0,
 '25%': 3.0,
 '50%': 3.0,
 '75%': 3.0,
 'max': 4.0},
'RelationshipSatisfaction': {'count': 1470.0,
 'mean': 2.7122448979591836,
 'std': 1.081208886440361,
 'min': 1.0,
 '25%': 2.0,
 '50%': 3.0,
 '75%': 4.0,
 'max': 4.0},
'StandardHours': {'count': 1470.0,
 'mean': 80.0,
 'std': 0.0,
 'min': 80.0,
 '25%': 80.0,
 '50%': 80.0,
 '75%': 80.0,
 'max': 80.0},
'StockOptionLevel': {'count': 1470.0,
 'mean': 0.7938775510204081,
 'std': 0.8520766679308365,
 'min': 0.0,
 '25%': 0.0,
 '50%': 1.0,
 '75%': 1.0,
 'max': 3.0},
'TotalWorkingYears': {'count': 1470.0,
 'mean': 11.279591836734694,
 'std': 7.780781675515004,
 'min': 0.0,
 '25%': 6.0,
 '50%': 10.0,
 '75%': 15.0,
 'max': 40.0},
'TrainingTimesLastYear': {'count': 1470.0,
 'mean': 2.7993197278911564,
 'std': 1.2892706207958435,
 'min': 0.0,
 '25%': 2.0,
 '50%': 3.0,
```

```
  '75%': 3.0,
  'max': 6.0},
 'WorkLifeBalance': {'count': 1470.0,
  'mean': 2.7612244897959184,
  'std': 0.7064758297141522,
  'min': 1.0,
  '25%': 2.0,
  '50%': 3.0,
  '75%': 3.0,
  'max': 4.0},
 'YearsAtCompany': {'count': 1470.0,
  'mean': 7.0081632653061225,
  'std': 6.126525152403566,
  'min': 0.0,
  '25%': 3.0,
  '50%': 5.0,
  '75%': 9.0,
  'max': 40.0},
 'YearsInCurrentRole': {'count': 1470.0,
  'mean': 4.229251700680272,
  'std': 3.6231370346706395,
  'min': 0.0,
  '25%': 2.0,
  '50%': 3.0,
  '75%': 7.0,
  'max': 18.0},
 'YearsSinceLastPromotion': {'count': 1470.0,
  'mean': 2.1877551020408164,
  'std': 3.222430279137968,
  'min': 0.0,
  '25%': 0.0,
  '50%': 1.0,
  '75%': 3.0,
  'max': 15.0},
 'YearsWithCurrManager': {'count': 1470.0,
  'mean': 4.12312925170068,
  'std': 3.5681361205404407,
  'min': 0.0,
  '25%': 2.0,
  '50%': 3.0,
  '75%': 7.0,
  'max': 17.0}},
'unique_values': {'Age': 43,
 'Attrition': 2,
 'BusinessTravel': 3,
 'DailyRate': 886,
 'Department': 3,
 'DistanceFromHome': 29,
 'Education': 5,
 'EducationField': 6,
 'EmployeeCount': 1,
 'EmployeeNumber': 1470,
 'EnvironmentSatisfaction': 4,
 'Gender': 2,
 'HourlyRate': 71,
 'JobInvolvement': 4,
```

```
 'JobLevel': 5,
 'JobRole': 9,
 'JobSatisfaction': 4,
 'MaritalStatus': 3,
 'MonthlyIncome': 1349,
 'MonthlyRate': 1427,
 'NumCompaniesWorked': 10,
 'Over18': 1,
 'OverTime': 2,
 'PercentSalaryHike': 15,
 'PerformanceRating': 2,
 'RelationshipSatisfaction': 4,
 'StandardHours': 1,
 'StockOptionLevel': 4,
 'TotalWorkingYears': 40,
 'TrainingTimesLastYear': 7,
 'WorkLifeBalance': 4,
 'YearsAtCompany': 37,
 'YearsInCurrentRole': 19,
 'YearsSinceLastPromotion': 16,
 'YearsWithCurrManager': 18},
'data_types': {'Age': dtype('int64'),
 'Attrition': dtype('O'),
 'BusinessTravel': dtype('O'),
 'DailyRate': dtype('int64'),
 'Department': dtype('O'),
 'DistanceFromHome': dtype('int64'),
 'Education': dtype('int64'),
 'EducationField': dtype('O'),
 'EmployeeCount': dtype('int64'),
 'EmployeeNumber': dtype('int64'),
 'EnvironmentSatisfaction': dtype('int64'),
 'Gender': dtype('O'),
 'HourlyRate': dtype('int64'),
 'JobInvolvement': dtype('int64'),
 'JobLevel': dtype('int64'),
 'JobRole': dtype('O'),
 'JobSatisfaction': dtype('int64'),
 'MaritalStatus': dtype('O'),
 'MonthlyIncome': dtype('int64'),
 'MonthlyRate': dtype('int64'),
 'NumCompaniesWorked': dtype('int64'),
 'Over18': dtype('O'),
 'OverTime': dtype('O'),
 'PercentSalaryHike': dtype('int64'),
 'PerformanceRating': dtype('int64'),
 'RelationshipSatisfaction': dtype('int64'),
 'StandardHours': dtype('int64'),
 'StockOptionLevel': dtype('int64'),
 'TotalWorkingYears': dtype('int64'),
 'TrainingTimesLastYear': dtype('int64'),
 'WorkLifeBalance': dtype('int64'),
 'YearsAtCompany': dtype('int64'),
 'YearsInCurrentRole': dtype('int64'),
 'YearsSinceLastPromotion': dtype('int64'),
 'YearsWithCurrManager': dtype('int64')}}
```

Initial Data Analysis Summary

```
Total Records: 1,470 rows
Duplicates: No duplicate records
Missing Values: No missing values
```

```
Potential Issues:
EmployeeCount and StandardHours are constant
Over18 is also constant
EmployeeNumber is unique for each employee
```

In [25]:
```
# Drop unnecessary columns
columns_to_drop = ["EmployeeCount", "StandardHours", "Over18", "EmployeeNumber"]
df_cleaned = df.drop(columns=columns_to_drop)
```

Exploratory Data Analysis EDA

In [26]:
```
# attrition distribution

fig = px.pie(df, names="Attrition", title="Employee Attrition Distribution", hole=0

# Show Plot
fig.show()
```

Numeric Feature Analysis

```
1️⃣ Age Distribution
2️⃣ Monthly Income Distribution
3️⃣ Years at Company
```

In [27]:
```
# Age Distribution
fig_age = px.histogram(
    df_cleaned, x="Age", nbins=30,
    title="Age Distribution of Employees",
    labels={"Age": "Employee Age"},
    color_discrete_sequence=["#636EFA"],
    text_auto= True
)
fig_age.show()
```

In [28]:
```
# Monthly Income Distribution
fig_income = px.histogram(
    df_cleaned, x="MonthlyIncome", nbins=30,
    title="Monthly Income Distribution",
    labels={"MonthlyIncome": "Monthly Income ($)"},
    color_discrete_sequence=["#EF553B"],
    text_auto= True
)
```

```
fig_income.show()
```

In [29]:
```
# Years at Company Distribution
fig_years = px.histogram(
    df_cleaned, x="YearsAtCompany", nbins=30,
    title="Years at Company Distribution",
    labels={"YearsAtCompany": "Years at Company"},
    color_discrete_sequence=["#00CC96"],
    text_auto= True
)
fig_years.show()
```

Categorical Feature Analysis

1️⃣ Attrition by Department
2️⃣ Attrition by Job Role
3️⃣ Attrition by Marital Status
4️⃣ Attrition by Overtime Work

In [30]:
```
# Attrition by Department
fig_dept = px.bar(
    df_cleaned.groupby("Department")["Attrition"].value_counts().unstack(),
    title="Attrition by Department",
    labels={"value": "Frequency", "Department": "Department"},
    barmode="group",
    color_discrete_sequence=["#636EFA", "#EF553B"],
    text_auto= True
)
fig_dept.show()
```

In [31]:
```
# Attrition by Job Role
fig_role = px.bar(
    df_cleaned.groupby("JobRole")["Attrition"].value_counts().unstack(),
    title="Attrition by Job Role",
    labels={"value": "Frequency", "JobRole": "Job Role"},
    barmode="group",
    color_discrete_sequence=["#636EFA", "#EF553B"],
    text_auto= True
)
fig_role.show()
```

In [32]:
```
# Attrition by Marital Status
fig_marital = px.bar(
    df_cleaned.groupby("MaritalStatus")["Attrition"].value_counts().unstack(),
    title="Attrition by Marital Status",
    labels={"value": "frequency", "MaritalStatus": "Marital Status"},
    barmode="group",
    color_discrete_sequence=["#636EFA", "#EF553B"],
    text_auto= True
)
fig_marital.show()
```

In [33]:
```
# Attrition by Overtime Work
fig_overtime = px.bar(
    df_cleaned.groupby("OverTime")["Attrition"].value_counts().unstack(),
    title="Attrition by Overtime Work",
    labels={"value": "frequency", "OverTime": "Overtime"},
    barmode="group",
    color_discrete_sequence=["#636EFA", "#EF553B"],
    text_auto= True
)
fig_overtime.show()
```

Correlation Analysis

1️⃣ Correlation Heatmap Shows relationships between numeric features
2️⃣ Attrition vs. Salary, Job Satisfaction, and Work-Life Balance

In [34]:
```
import plotly.figure_factory as ff
import numpy as np

# Select only important numeric columns for attrition
important_features = [
    "Age", "MonthlyIncome", "TotalWorkingYears", "YearsAtCompany",
    "JobSatisfaction", "WorkLifeBalance"
]

df_selected = df_cleaned[important_features]

# Compute correlation matrix
correlation_matrix = df_selected.corr()

# Format numbers to 2 decimal places
z_text = np.around(correlation_matrix.values, decimals=2).astype(str)

# Create heatmap
fig_corr = ff.create_annotated_heatmap(
    z=correlation_matrix.values,
    x=list(correlation_matrix.columns),
    y=list(correlation_matrix.index),
    annotation_text=z_text,  # Add formatted text
    colorscale="Blues",
    showscale=True
)

# Set title
fig_corr.update_layout(title_text="Correlation Heatmap (Key Features)")

# Show figure
fig_corr.show()
```

In [35]:
```
# Attrition vs. Monthly Income
fig_income_attrition = px.box(
    df_cleaned, x="Attrition", y="MonthlyIncome",
```

```
    color="Attrition",
    title="Attrition vs. Monthly Income",
    labels={"MonthlyIncome": "Monthly Income ($)", "Attrition": "Attrition Status"}
    color_discrete_sequence=["#636EFA", "#EF553B"],

)
fig_income_attrition.show()
```

In [36]:
```
# Attrition vs. Job Satisfaction
fig_job_satisfaction = px.histogram(
    df_cleaned, x="JobSatisfaction", color="Attrition",
    title="Attrition vs. Job Satisfaction",
    labels={"JobSatisfaction": "Job Satisfaction Level", "Attrition": "Attrition St
    barmode="group",
    color_discrete_sequence=["#636EFA", "#EF553B"],
    text_auto= True
)
fig_job_satisfaction.show()
```

In [37]:
```
# Attrition vs. Work-Life Balance
fig_work_life = px.histogram(
    df_cleaned, x="WorkLifeBalance", color="Attrition",
    title="Attrition vs. Work-Life Balance",
    labels={"WorkLifeBalance": "Work-Life Balance Level", "Attrition": "Attrition S
    barmode="group",
    color_discrete_sequence=["#EF553B", "#636EFA"],
        text_auto= True
)
fig_work_life.show()
```

Predictive Model for Employee Attrition

In [38]:
```
# Encode categorical variables
df_encoded = df_cleaned.copy()
label_encoders = {}

for col in df_encoded.select_dtypes(include=["object"]).columns:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
    label_encoders[col] = le
```

In [39]:
```
# Define features and target variable
X = df_encoded.drop(columns=["Attrition"])  # Features
y = df_encoded["Attrition"]  # Target variable
```

In [40]:
```
# Split data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [41]:
```
# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [42]:
```python
# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

Out[42]:
```
▾ LogisticRegression

LogisticRegression()
```

In [43]:
```python
# Make predictions
y_pred = model.predict(X_test_scaled)
```

In [44]:
```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

accuracy, report
```

Out[44]:
```
(0.8945578231292517,
 '              precision    recall  f1-score   support\n\n           0       0.91
0.98      0.94       255\n           1       0.70      0.36      0.47        39\n
\n    accuracy                           0.89       294\n   macro avg       0.80
0.67      0.71       294\nweighted avg       0.88      0.89      0.88       294
\n')
```

✅ Model Results

Accuracy: 89.5%
Precision & Recall:
Employees Staying (0): 91% precision, 98% recall
Employees Leaving (1): 70% precision, 36% recall
🔹 Interpretation: The model predicts employees staying well but struggles with employees leaving (attrition cases are fewer).