In [2]:
```
# Load Libraries

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

In [4]:
```
#Load dataset
data = pd.read_csv("ibm_hr_employee-attrition.csv")
print("Dataset loaded successfully.")
data.info()
```

```
Dataset loaded successfully.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [18]:
```
# Identify column types
num_columns = data.select_dtypes(include=['number']).columns
cat_columns = data.select_dtypes(exclude=['number']).columns

print(f"\n🔢 Numerical columns ({len(num_columns)}):\n{list(num_columns)}")
print(f"\n🔤 Categorical columns ({len(cat_columns)}):\n{list(cat_columns)}")
```

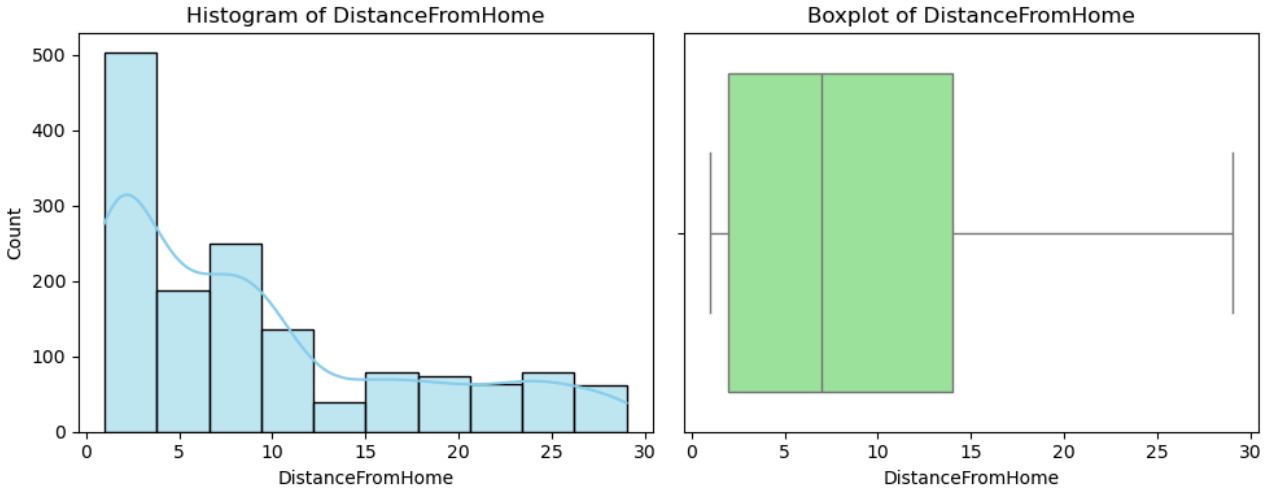
```
🔢 Numerical columns (26):
['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount', 'EmployeeNumber', 'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'J

🔤 Categorical columns (9):
['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Over18', 'OverTime']
```

In [22]:
```
# Prompt user for a numerical column
num_column = input("\nEnter the name of a numerical column: ")

if num_column in data.columns:
    column_data = data[num_column]

    # Descriptive statistics
    print(f"\n📊 Statistics for '{num_column}':\n")
    print(f"Mean: {column_data.mean():.2f}")
    print(f"Median: {column_data.median():.2f}")
    print(f"Mode: {column_data.mode().iloc[0] if not column_data.mode().empty else 'N/A'}")
    print(f"Standard Deviation: {column_data.std():.2f}")
    print(f"Variance: {column_data.var():.2f}")
    print(f"Range: {column_data.max() - column_data.min():.2f}")

    # Histogram and Boxplot
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 1)
    sns.histplot(column_data, kde=True, bins=10, color='skyblue')
    plt.title(f"Histogram of {num_column}")

    plt.subplot(1, 2, 2)
    sns.boxplot(x=column_data, color='lightgreen')
    plt.title(f"Boxplot of {num_column}")

    plt.tight_layout()
    plt.show()

    # Outlier Detection
    q1 = column_data.quantile(0.25)
    q3 = column_data.quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
```

```
    outliers = column_data[(column_data < lower) | (column_data > upper)]
    print(f"\n🚨 Number of outliers in '{num_column}': {len(outliers)}")
    if not outliers.empty:
        display(outliers)
else:
    print(f"⚠️ Column '{num_column}' not found in the dataset.")
```

```
Enter the name of a numerical column:  DistanceFromHome
```

```
📊 Statistics for 'DistanceFromHome':

Mean: 9.19
Median: 7.00
Mode: 2
Standard Deviation: 8.11
Variance: 65.72
Range: 28.00
```

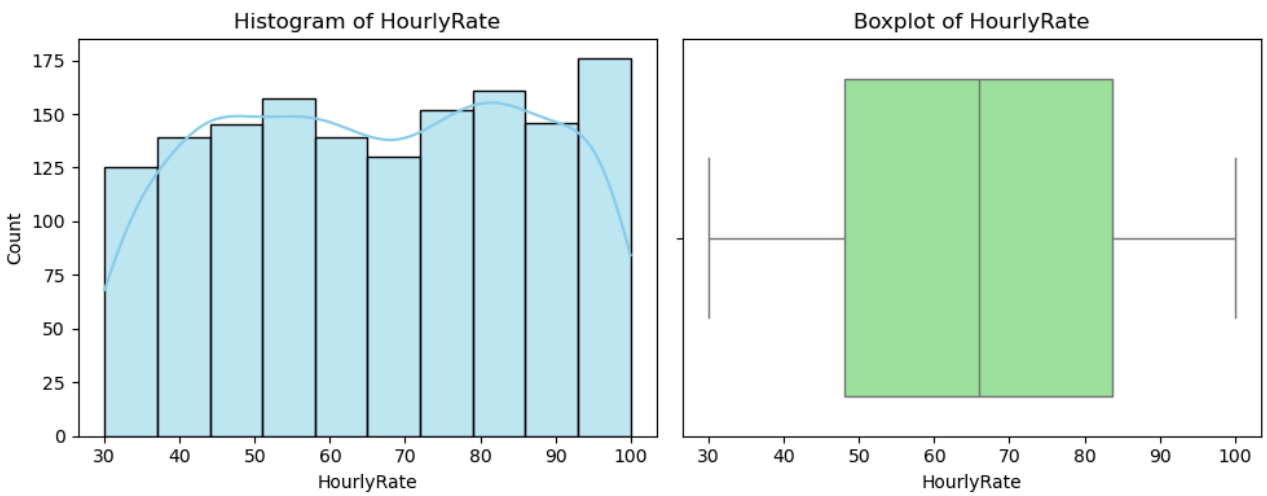



```
🚨 Number of outliers in 'DistanceFromHome': 0
```

In [28]:
```
# Prompt user for a numerical column
num_column = input("\nEnter the name of a numerical column: ")

if num_column in data.columns:
    column_data = data[num_column]

    # Descriptive statistics
    print(f"\n📊 Statistics for '{num_column}':\n")
    print(f"Mean: {column_data.mean():.2f}")
    print(f"Median: {column_data.median():.2f}")
    print(f"Mode: {column_data.mode().iloc[0] if not column_data.mode().empty else 'N/A'}")
    print(f"Standard Deviation: {column_data.std():.2f}")
    print(f"Variance: {column_data.var():.2f}")
    print(f"Range: {column_data.max() - column_data.min():.2f}")

    # Histogram and Boxplot
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 1)
    sns.histplot(column_data, kde=True, bins=10, color='skyblue')
    plt.title(f"Histogram of {num_column}")

    plt.subplot(1, 2, 2)
    sns.boxplot(x=column_data, color='lightgreen')
    plt.title(f"Boxplot of {num_column}")

    plt.tight_layout()
    plt.show()

    # Outlier Detection
    q1 = column_data.quantile(0.25)
    q3 = column_data.quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr

    outliers = column_data[(column_data < lower) | (column_data > upper)]
    print(f"\n🚨 Number of outliers in '{num_column}': {len(outliers)}")
    if not outliers.empty:
        display(outliers)
else:
    print(f"⚠️ Column '{num_column}' not found in the dataset.")
```

```
Enter the name of a numerical column:  HourlyRate
```

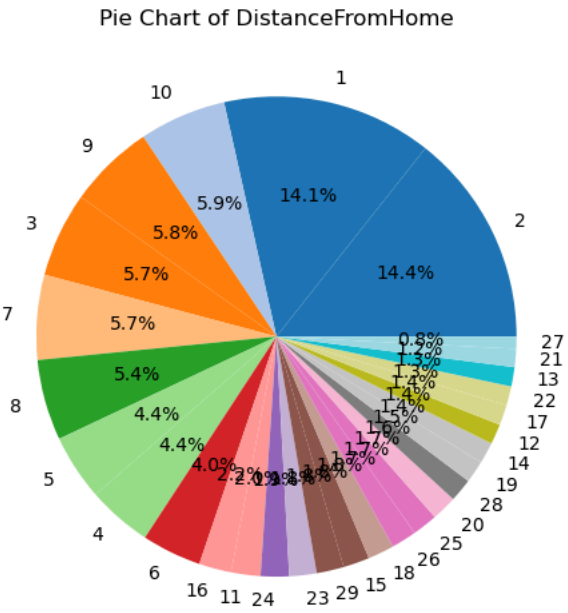
```
📊 Statistics for 'HourlyRate':

Mean: 65.89
Median: 66.00
Mode: 66
Standard Deviation: 20.33
Variance: 413.29
Range: 70.00
```



🚨 Number of outliers in 'HourlyRate': 0

In [32]:
```
# Prompt user for a categorical column
cat_column = input("\nEnter the name of a categorical column: ")

if cat_column in data.columns:
    category_counts = data[cat_column].value_counts()
    chart_type = input("Choose chart type (bar/pie): ").lower()

    if chart_type == 'bar':
        category_counts.plot(kind='bar', color='orange', figsize=(6, 3))
        plt.title(f"Bar Chart of {cat_column}")
        plt.xlabel(cat_column)
        plt.ylabel("Frequency")
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

    elif chart_type == 'pie':
        category_counts.plot(kind='pie', autopct="%.1f%%", figsize=(5, 5), colormap='tab20')
        plt.title(f"Pie Chart of {cat_column}")
        plt.ylabel("")
        plt.tight_layout()
        plt.show()

    else:
        print("⚠️ Invalid chart type. Please choose either 'bar' or 'pie'.")

    print(f"\n📋 Frequency of categories in '{cat_column}':\n")
    display(category_counts)
else:
    print(f"⚠️ Column '{cat_column}' not found in the dataset.")
```

```
Enter the name of a categorical column:  DistanceFromHome
Choose chart type (bar/pie):  pie
```



📋 Frequency of categories in 'DistanceFromHome':

```
DistanceFromHome
2      211
1      208
10      86
9       85
3       84
```

```
7      84
8      80
5      65
4      64
6      59
16     32
11     29
24     28
23     27
29     27
15     26
18     26
26     25
25     25
20     25
28     23
19     22
14     21
12     20
17     20
22     19
13     19
21     18
27     12
Name: count, dtype: int64
```