```
#Rishikesh_2412res99


# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the data from CSV file
data = pd.read_csv("/content/HR_comma_sep.csv")

# Quick look at the data
print(data.head())
```

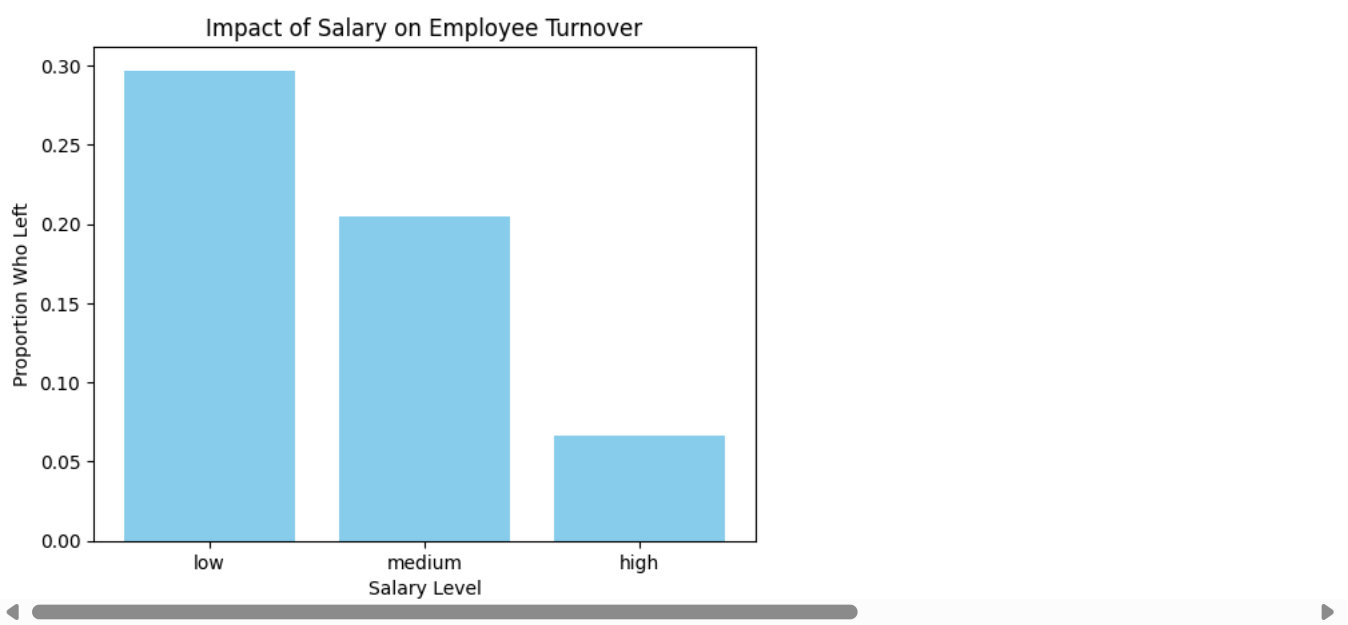
```
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157
1                0.80             0.86               5                   262
2                0.11             0.88               7                   272
3                0.72             0.87               5                   223
4                0.37             0.52               2                   159

   time_spend_company  Work_accident  left  promotion_last_5years Department  \
0                   3              0     1                      0      sales
1                   6              0     1                      0      sales
2                   4              0     1                      0      sales
3                   5              0     1                      0      sales
4                   3              0     1                      0      sales

   salary
0     low
1  medium
2  medium
3     low
4     low
```

```
# Split data into employees who left (1) and stayed (0)
left = data[data["left"] == 1]
stayed = data[data["left"] == 0]

# Calculate means for key variables
print("Employees who LEFT:")
print(left[["satisfaction_level", "last_evaluation", "number_project",
           "average_montly_hours", "time_spend_company", "Work_accident",
           "promotion_last_5years"]].mean())

print("\nEmployees who STAYED:")
print(stayed[["satisfaction_level", "last_evaluation", "number_project",
              "average_montly_hours", "time_spend_company", "Work_accident",
              "promotion_last_5years"]].mean())

# Observations (you can read these in output):
# - Low satisfaction, extreme projects/hours, mid-tenure, no promotions linked to leaving
```

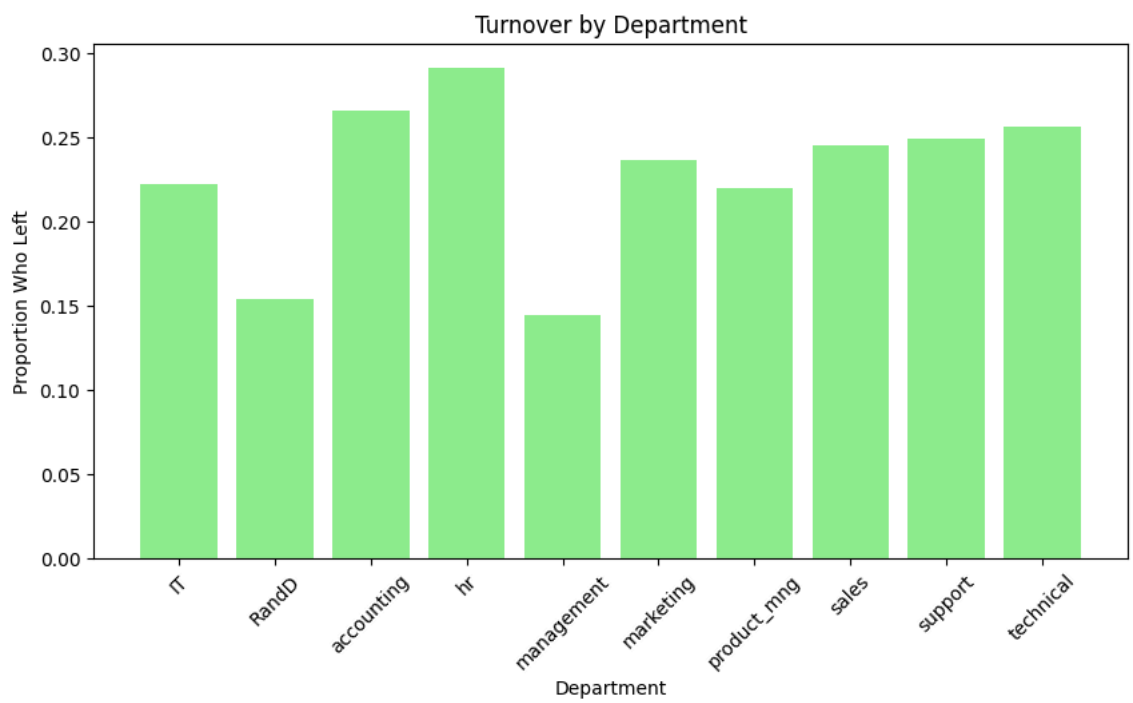
```
Employees who LEFT:
satisfaction_level         0.440098
last_evaluation            0.718113
number_project             3.855503
average_montly_hours     207.419210
time_spend_company         3.876505
Work_accident              0.047326
promotion_last_5years      0.005321
dtype: float64

Employees who STAYED:
satisfaction_level         0.666810
last_evaluation            0.715473
number_project             3.786664
average_montly_hours     199.060203
time_spend_company         3.380032
Work_accident              0.175009
promotion_last_5years      0.026251
dtype: float64
```

```
# Group by salary and calculate proportion who left
salary_retention = data.groupby("salary")["left"].mean()

# Order salary levels: low, medium, high
salary_retention = salary_retention.reindex(["low", "medium", "high"])

# Plot bar chart
plt.bar(salary_retention.index, salary_retention.values, color="skyblue")
```

```
plt.title("Impact of Salary on Employee Turnover")
plt.xlabel("Salary Level")
plt.ylabel("Proportion Who Left")
plt.show()
```




```
# Group by department and calculate proportion who left
dept_retention = data.groupby("Department")["left"].mean()

# Plot bar chart
plt.figure(figsize=(10, 5))  # Make it wider to fit all departments
plt.bar(dept_retention.index, dept_retention.values, color="lightgreen")
plt.title("Turnover by Department")
plt.xlabel("Department")
plt.ylabel("Proportion Who Left")
plt.xticks(rotation=45)  # Rotate labels for readability
plt.show()
```




```
# Prepare the data
# Convert salary to numbers (low=0, medium=1, high=2)
data["salary_num"] = data["salary"].map({"low": 0, "medium": 1, "high": 2})

# Select features (X) and target (y)
features = ["satisfaction_level", "number_project", "average_montly_hours",
            "time_spend_company", "promotion_last_5years", "salary_num"]
X = data[features]
y = data["left"]

# Split data into training (70%) and testing (30%)
```

```
# Split data into training (70%) and testing (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and train the model
model = LogisticRegression(max_iter=1000)  # max_iter avoids convergence warning
model.fit(X_train, y_train)

print("Model trained successfully!")
```

```
Model trained successfully!
```

```
# Make predictions on test data
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f} ({accuracy * 100:.1f}%)")
```

```
Model Accuracy: 0.77 (76.6%)
```