# Data Generation

## Chat-GPT Prompts

Generate python script to generate a realistic dataset of 8950 records for human resources. The dataset should include the following attributes:

1. Employee ID: A unique identifier.
2. First Name: Randomly generated.
3. Last Name: Randomly generated.
4. Gender: Randomly chosen with a 46% probability for ‘Female’ and a 54% probability for ‘Male’.
5. State and City: Randomly assigned from a predefined list of states and their cities.
6. 6.Hire Date: Randomly generated with custom probabilities for each year from 2015 to 2024.
7. 7.Department: Randomly chosen from a list of departments with specified probabilities.
8. Job Title: Randomly selected based on the department, with specific probabilities for each job title within the department.
9. Education Level: Determined based on the job title, chosen from a predefined mapping of job titles to education levels.
10. Performance Rating: Randomly selected from ‘Excellent’, ‘Good’, ‘Satisfactory’, ‘Needs Improvement’ with specified probabilities.
11. Overtime: Randomly chosen with a 30% probability for ‘Yes’ and a 70% probability for ‘No’.
12. Salary: Generated based on the department and job title, within specific ranges.
13. Birth Date: Generated based on age group distribution and job title requirements, ensuring consistency with the hire date.
14. Termination Date: Assigned to a subset of employees (11.2% of the total) with specific probabilities for each year from 2015 to 2024, ensuring the termination date is at least 6 months after the hire date.
15. Adjusted Salary: Calculated based on gender, education level, and age, applying specific multipliers and increments.
16. Be sure to structure the code cleanly, using functions where appropriate, and include comments to explain each step of the process.

## Python Script

import pandas as pd

import numpy as np

from faker import Faker

from datetime import datetime, timedelta

import random

# Initialize Faker

fake = Faker('en\_US')

Faker.seed(42)

np.random.seed(42)

random.seed(42)

# Configuration

num\_records = 8950

# States & Cities

states\_cities = {

'New York': ['New York City', 'Buffalo', 'Rochester'],

'Virginia': ['Virginia Beach', 'Norfolk', 'Richmond'],

'Florida': ['Miami', 'Orlando', 'Tampa'],

'Illinois': ['Chicago', 'Aurora', 'Naperville'],

'Pennsylvania': ['Philadelphia', 'Pittsburgh', 'Allentown'],

'Ohio': ['Columbus', 'Cleveland', 'Cincinnati'],

'North Carolina': ['Charlotte', 'Raleigh', 'Greensboro'],

'Michigan': ['Detroit', 'Grand Rapids', 'Warren']

}

states = list(states\_cities.keys())

state\_prob = [0.7, 0.02, 0.01, 0.03, 0.05, 0.03, 0.05, 0.11]

assigned\_states = np.random.choice(states, size=num\_records, p=state\_prob)

assigned\_cities = [np.random.choice(states\_cities[state]) for state in assigned\_states]

# Departments & Jobtitles

departments = ['HR', 'IT', 'Sales', 'Marketing', 'Finance', 'Operations', 'Customer Service']

departments\_prob = [0.02, 0.15, 0.21, 0.08, 0.05, 0.30, 0.19]

jobtitles = {

'HR': ['HR Manager', 'HR Coordinator', 'Recruiter', 'HR Assistant'],

'IT': ['IT Manager', 'Software Developer', 'System Administrator', 'IT Support Specialist'],

'Sales': ['Sales Manager', 'Sales Consultant', 'Sales Specialist', 'Sales Representative'],

'Marketing': ['Marketing Manager', 'SEO Specialist', 'Content Creator', 'Marketing Coordinator'],

'Finance': ['Finance Manager', 'Accountant', 'Financial Analyst', 'Accounts Payable Specialist'],

'Operations': ['Operations Manager', 'Operations Analyst', 'Logistics Coordinator', 'Inventory Specialist'],

'Customer Service': ['Customer Service Manager', 'Customer Service Representative', 'Support Specialist', 'Help Desk Technician']

}

jobtitles\_prob = {

'HR': [0.03, 0.3, 0.47, 0.2], # HR Manager, HR Coordinator, Recruiter, HR Assistant

'IT': [0.02, 0.47, 0.2, 0.31], # IT Manager, Software Developer, System Administrator, IT Support Specialist

'Sales': [0.03, 0.25, 0.32, 0.4], # Sales Manager, Sales Consultant, Sales Specialist, Sales Representative

'Marketing': [0.04, 0.25, 0.41, 0.3], # Marketing Manager, SEO Specialist, Content Creator, Marketing Coordinator

'Finance': [0.03, 0.37, 0.4, 0.2], # Finance Manager, Accountant, Financial Analyst, Accounts Payable Specialist

'Operations': [0.02, 0.2, 0.4, 0.38], # Operations Manager, Operations Analyst, Logistics Coordinator, Inventory Specialist

'Customer Service': [0.04, 0.3, 0.38, 0.28] # Customer Service Manager, Customer Service Representative, Support Specialist, Help Desk Technician

}

# Educations

educations = ['High School', "Bachelor", "Master", 'PhD']

education\_mapping = {

'HR Manager': ["Master", "PhD"],

'HR Coordinator': ["Bachelor", "Master"],

'Recruiter': ["High School", "Bachelor"],

'HR Assistant': ["High School", "Bachelor"],

'IT Manager': ["PhD", "Master"],

'Software Developer': ["Bachelor", "Master"],

'System Administrator': ["Bachelor", "Master"],

'IT Support Specialist': ["High School", "Bachelor"],

'Sales Manager': ["Master","PhD"],

'Sales Consultant': ["Bachelor", "Master", "PhD"],

'Sales Specialist': ["Bachelor", "Master", "PhD"],

'Sales Representative': ["Bachelor"],

'Marketing Manager': ["Bachelor", "Master","PhD"],

'SEO Specialist': ["High School", "Bachelor"],

'Content Creator': ["High School", "Bachelor"],

'Marketing Coordinator': ["Bachelor"],

'Finance Manager': ["Master", "PhD"],

'Accountant': ["Bachelor"],

'Financial Analyst': ["Bachelor", "Master", "PhD"],

'Accounts Payable Specialist': ["Bachelor"],

'Operations Manager': ["Bachelor", "Master"],

'Operations Analyst': ["Bachelor", "Master"],

'Logistics Coordinator': ["Bachelor"],

'Inventory Specialist': ["High School", "Bachelor"],

'Customer Service Manager': ["Bachelor", "Master", "PhD"],

'Customer Service Representative': ["High School", "Bachelor"],

'Support Specialist': ["High School", "Bachelor"],

'Customer Success Manager': ["Bachelor", "Master", "PhD"],

'Help Desk Technician': ["High School", "Bachelor"]

}

# Hiring Date

# Define custom probability weights for each year

year\_weights = {

2015: 5, # 15% probability

2016: 8, # 15% probability

2017: 17, # 20% probability

2018: 9, # 15% probability

2019: 10, # 10% probability

2020: 11, # 10% probability

2021: 5, # 8% probability

2022: 12, # 5% probability

2023: 14, # 2% probability

2024: 9 # 2% probability

}

# Generate a random date based on custom probabilities

def generate\_custom\_date(year\_weights):

year = random.choices(list(year\_weights.keys()), weights=list(year\_weights.values()))[0]

month = random.randint(1, 12)

day = random.randint(1, 28) # Assuming all months have 28 days for simplicity

return fake.date\_time\_between(start\_date=datetime(year, 1, 1), end\_date=datetime(year, 12, 31))

def generate\_salary(department, job\_title):

salary\_dict = {

'HR': {

'HR Manager': np.random.randint(60000, 90000),

'HR Coordinator': np.random.randint(50000, 60000),

'Recruiter': np.random.randint(50000, 70000),

'HR Assistant': np.random.randint(50000, 60000)

},

'IT': {

'IT Manager': np.random.randint(80000, 120000),

'Software Developer': np.random.randint(70000, 95000),

'System Administrator': np.random.randint(60000, 90000),

'IT Support Specialist': np.random.randint(50000, 60000)

},

'Sales': {

'Sales Manager': np.random.randint(70000, 110000),

'Sales Consultant': np.random.randint(60000, 90000),

'Sales Specialist': np.random.randint(50000, 80000),

'Sales Representative': np.random.randint(50000, 70000)

},

'Marketing': {

'Marketing Manager': np.random.randint(70000, 100000),

'SEO Specialist': np.random.randint(50000, 80000),

'Content Creator': np.random.randint(50000, 60000),

'Marketing Coordinator': np.random.randint(50000, 70000)

},

'Finance': {

'Finance Manager': np.random.randint(80000, 120000),

'Accountant': np.random.randint(50000, 80000),

'Financial Analyst': np.random.randint(60000, 90000),

'Accounts Payable Specialist': np.random.randint(50000, 60000)

},

'Operations': {

'Operations Manager': np.random.randint(70000, 100000),

'Operations Analyst': np.random.randint(50000, 80000),

'Logistics Coordinator': np.random.randint(50000, 60000),

'Inventory Specialist': np.random.randint(50000, 60000)

},

'Customer Service': {

'Customer Service Manager': np.random.randint(60000, 90000),

'Customer Service Representative': np.random.randint(50000, 60000),

'Support Specialist': np.random.randint(50000, 60000),

'Help Desk Technician': np.random.randint(50000, 80000)

}

}

return salary\_dict[department][job\_title]

# Generate the dataset

data = []

for \_ in range(num\_records):

employee\_id = f"00-{random.randint(10000000, 99999999)}"

first\_name = fake.first\_name()

last\_name = fake.last\_name()

gender = np.random.choice(['Female', 'Male'], p=[0.46, 0.54])

state = np.random.choice(states, p=state\_prob)

city = np.random.choice(states\_cities[state])

hiredate = generate\_custom\_date(year\_weights)

#termdate

department = np.random.choice(departments, p=departments\_prob)

job\_title = np.random.choice(jobtitles[department], p=jobtitles\_prob[department])

education\_level = np.random.choice(education\_mapping[job\_title])

performance\_rating = np.random.choice(['Excellent', 'Good', 'Satisfactory', 'Needs Improvement'], p=[0.12, 0.5, 0.3, 0.08])

overtime = np.random.choice(['Yes', 'No'], p=[0.3, 0.7])

salary = generate\_salary(department, job\_title)

data.append([

employee\_id,

first\_name,

last\_name,

gender,

state,

city,

hiredate,

department,

job\_title,

education\_level,

salary,

performance\_rating,

overtime

])

## Create DataFrame

columns = [

'employee\_id',

'first\_name',

'last\_name',

'gender',

'state',

'city',

'hiredate',

'department',

'job\_title',

'education\_level',

'salary',

'performance\_rating',

'overtime'

]

df = pd.DataFrame(data, columns=columns)

# Add Birthdate

def generate\_birthdate(row):

age\_distribution = {

'under\_25': 0.11,

'25\_34': 0.25,

'35\_44': 0.31,

'45\_54': 0.24,

'over\_55': 0.09

}

age\_groups = list(age\_distribution.keys())

age\_probs = list(age\_distribution.values())

age\_group = np.random.choice(age\_groups, p=age\_probs)

if any('Manager' in title for title in row['job\_title']):

age = np.random.randint(30, 65)

elif row['education\_level'] == 'PhD':

age = np.random.randint(27, 65)

elif age\_group == 'under\_25':

age = np.random.randint(20, 25)

elif age\_group == '25\_34':

age = np.random.randint(25, 35)

elif age\_group == '35\_44':

age = np.random.randint(35, 45)

elif age\_group == '45\_54':

age = np.random.randint(45, 55)

else:

age = np.random.randint(56, 65)

birthdate = fake.date\_of\_birth(minimum\_age=age, maximum\_age=age)

return birthdate

# Apply the function to generate birthdates

df['birthdate'] = df.apply(generate\_birthdate, axis=1)

# Terminations

# Define termination distribution

year\_weights = {

2015: 5,

2016: 7,

2017: 10,

2018: 12,

2019: 9,

2020: 10,

2021: 20,

2022: 10,

2023: 7,

2024: 10

}

# Calculate the total number of terminated employees

total\_employees = num\_records

termination\_percentage = 0.112 # 11.2%

total\_terminated = int(total\_employees \* termination\_percentage)

# Generate termination dates based on distribution

termination\_dates = []

for year, weight in year\_weights.items():

num\_terminations = int(total\_terminated \* (weight / 100))

termination\_dates.extend([year] \* num\_terminations)

# Randomly shuffle the termination dates

random.shuffle(termination\_dates)

# Assign termination dates to terminated employees

terminated\_indices = df.index[:total\_terminated]

for i, year in enumerate(termination\_dates[:total\_terminated]):

df.at[terminated\_indices[i], 'termdate'] = datetime(year, 1, 1) + timedelta(days=random.randint(0, 365))

# Assign None to termdate for employees who are not terminated

df['termdate'] = df['termdate'].where(df['termdate'].notnull(), None)

# Ensure termdate is at least 6 months after hiredat

df['termdate'] = df.apply(lambda row: row['hiredate'] + timedelta(days=180) if row['termdate'] and row['termdate'] < row['hiredate'] + timedelta(days=180) else row['termdate'], axis=1)

education\_multiplier = {

'High School': {'Male': 1.03, 'Female': 1.0},

"Bachelor": {'Male': 1.115, 'Female': 1.0},

"Master": {'Male': 1.0, 'Female': 1.07},

'PhD': {'Male': 1.0, 'Female': 1.17}

}

# Function to calculate age from birthdate

def calculate\_age(birthdate):

today = pd.Timestamp('today')

age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))

return age

# Function to calculate the adjusted salary

def calculate\_adjusted\_salary(row):

base\_salary = row['salary']

gender = row['gender']

education = row['education\_level']

age = calculate\_age(row['birthdate'])

# Apply education multiplier

multiplier = education\_multiplier.get(education, {}).get(gender, 1.0)

adjusted\_salary = base\_salary \* multiplier

# Apply age increment (between 0.1% and 0.3% per year of age)

age\_increment = 1 + np.random.uniform(0.001, 0.003) \* age

adjusted\_salary \*= age\_increment

# Ensure the adjusted salary is not lower than the base salary

adjusted\_salary = max(adjusted\_salary, base\_salary)

# Round the adjusted salary to the nearest integer

return round(adjusted\_salary)

# Apply the function to the DataFrame

df['salary'] = df.apply(calculate\_adjusted\_salary, axis=1)

# Convert 'hiredate' and 'birthdate' to datetime

df['hiredate'] = pd.to\_datetime(df['hiredate']).dt.date

df['birthdate'] = pd.to\_datetime(df['birthdate']).dt.date

df['termdate'] = pd.to\_datetime(df['termdate']).dt.date

print(df)

# Save to CSV

df.to\_csv('HumanResources.csv', index=False)