



Home > Tableau > **Tableau HR Project Thank You**

 Search...

Tableau Project - HR Dashboard

This Tableau project is a step-by-step learning experience in building dashboard projects using Tableau from requirements to professional dashboard like I do in my real-world projects.

Project Download Materials

Links to Course Resources

- Download **Project Materials**
- Link to **Tableau Git** Repository
- Link to **Tableau Public** "Data With Baraa"

The Zip folder contains the following:

- **Project Data** – The data used in this HR Dashboard project is generated using a combination of ChatGPT prompts and the Python **Faker library**. This dataset simulates a set of employee information typically found in HR systems, including demographics, job details, salary, performance evaluations, and attrition data. The generated data is designed to mimic real-world HR data, providing a rich dataset for analysis and visualization in Tableau.
- **Icons & Images** – The icons used in the HR Dashboard are sourced from **Flaticon** and customized using **Photopea** to match the dashboard's color scheme. The PDS (Photopea files) can be found in the icon folder of the zip file for further editing if needed.
- **Mockups** – The initial dashboard mockup was created using the Procreate app on a tablet. Additionally, the dashboard container mockups were created using **draw.io**.

- **Tableau Project File** – The project file created during the course can be found in Zip file. You can also directly download the Tableau file from my [Tableau Public Profile](#).

!! IMPORTANT !!

Feel free to download and use the data provided in this tutorial. If you decided to share the Tableau project to Tableau public or in [LinkedIn](#), it would be appreciate it, if you mention [my youtube channel](#).

Instead of downloading the Tableau file directly, I suggest following along with me step by step. This way, you'll learn Tableau and have a feeling about the progress of projects and building dashboards in Tableau.

Other Links



[FIGMA – HR Dashboard Background Design](#)

[Download Tableau Public | Guide: How to Download Tableau Public](#)

[Tableau Public Account | Guide: How to Create Tableau Account](#)

[Download Draw.io](#)

User Story - HR Dashboard

As an HR manager, I want a comprehensive dashboard to analyze human resources data, providing both summary views for high-level insights and detailed employee records for in-depth analysis

Summary View

The summary view should be divided into three main sections: Overview, Demographics, and Income Analysis

Overview

The Overview section should provide a snapshot of the overall HR metrics, including:

- Display the total number of hired employees, active employees, and terminated employees.
- Visualize the total number of hired and terminated employees over the years.
- Present a breakdown of total employees by department and job titles.
- Compare total employees between headquarters (HQ) and branches (New York is the HQ)
- Show the distribution of employees by city and state.

Demographics

The Demographics section should offer insights into the composition of the workforce, including:

- Present the gender ratio in the company.
- Visualize the distribution of employees across age groups and education levels.
- Show the total number of employees within each age group.
- Show the total number of employees within each education level.
- Present the correlation between employees's educational backgrounds and their performance ratings.

Income

The income analysis section should focus on salary-related metrics, including:

- Compare salaries across different education levels for both genders to identify any discrepancies or patterns.

- Present how the age correlate with the salary for employees in each department.

Employee Records View

- Provide a comprehensive list of all employees with necessary information such as name, department, position, gender, age, education, and salary.
 - Users should be able to filter the list based on any of the available columns.
-

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. Hire Date: Randomly generated with custom probabilities for each year from 2015 to 2024.
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

```

1  import pandas as pd
2  import numpy as np
3  from faker import Faker
4  from datetime import datetime, timedelta
5  import random
6
7  # Initialize Faker
8  fake = Faker('en_US')
9  Faker.seed(42)
10 np.random.seed(42)
11 random.seed(42)
12
13 # Configuration
14 num_records = 8950
15
16 # States & Cities
17 states_cities = {
18     'New York': ['New York City', 'Buffalo', 'Rochester'],
19     'Virginia': ['Virginia Beach', 'Norfolk', 'Richmond'],
20     'Florida': ['Miami', 'Orlando', 'Tampa'],
21     'Illinois': ['Chicago', 'Aurora', 'Naperville'],
22     'Pennsylvania': ['Philadelphia', 'Pittsburgh', 'Allentown'],
23     'Ohio': ['Columbus', 'Cleveland', 'Cincinnati'],
24     'North Carolina': ['Charlotte', 'Raleigh', 'Greensboro'],
25     'Michigan': ['Detroit', 'Grand Rapids', 'Warren']
26 }
27
28 states = list(states_cities.keys())
29 state_prob = [0.7, 0.02, 0.01, 0.03, 0.05, 0.03, 0.05, 0.11]
30 assigned_states = np.random.choice(states, size=num_records, p=state_prob)
31 assigned_cities = [np.random.choice(states_cities[state]) for state in
32 assigned_states]
33
34 # Departments & Jobtitles
35 departments = ['HR', 'IT', 'Sales', 'Marketing', 'Finance', 'Operations',
36 'Customer Service']
37 departments_prob = [0.02, 0.15, 0.21, 0.08, 0.05, 0.30, 0.19]
38 jobtitles = {
39     'HR': ['HR Manager', 'HR Coordinator', 'Recruiter', 'HR Assistant'],
40     'IT': ['IT Manager', 'Software Developer', 'System Administrator', 'IT
41 Support Specialist'],
42     'Sales': ['Sales Manager', 'Sales Consultant', 'Sales Specialist', 'Sales
43 Representative'],
44     'Marketing': ['Marketing Manager', 'SEO Specialist', 'Content Creator',
45 'Marketing Coordinator'],
46     'Finance': ['Finance Manager', 'Accountant', 'Financial Analyst', 'Accounts
47 Payable Specialist'],
48     'Operations': ['Operations Manager', 'Operations Analyst', 'Logistics

```

```

43 Coordinator', 'Inventory Specialist'],
44     'Customer Service': ['Customer Service Manager', 'Customer Service
45 Representative', 'Support Specialist', 'Help Desk Technician']
46 }
47 jobtitles_prob = {
48     'HR': [0.03, 0.3, 0.47, 0.2], # HR Manager, HR Coordinator, Recruiter, HR
49 Assistant
50     'IT': [0.02, 0.47, 0.2, 0.31], # IT Manager, Software Developer, System
Administrator, IT Support Specialist
51     'Sales': [0.03, 0.25, 0.32, 0.4], # Sales Manager, Sales Consultant, Sales
Specialist, Sales Representative
52     'Marketing': [0.04, 0.25, 0.41, 0.3], # Marketing Manager, SEO Specialist,
Content Creator, Marketing Coordinator
53     'Finance': [0.03, 0.37, 0.4, 0.2], # Finance Manager, Accountant, Financial
54 Analyst, Accounts Payable Specialist
55     'Operations': [0.02, 0.2, 0.4, 0.38], # Operations Manager, Operations
56 Analyst, Logistics Coordinator, Inventory Specialist
57     'Customer Service': [0.04, 0.3, 0.38, 0.28] # Customer Service Manager,
58 Customer Service Representative, Support Specialist, Help Desk Technician
59 }
60
61 # Educations
62 educations = ['High School', "Bachelor", "Master", 'PhD']
63
64 education_mapping = {
65     'HR Manager': ["Master", "PhD"],
66     'HR Coordinator': ["Bachelor", "Master"],
67     'Recruiter': ["High School", "Bachelor"],
68     'HR Assistant': ["High School", "Bachelor"],
69     'IT Manager': ["PhD", "Master"],
70     'Software Developer': ["Bachelor", "Master"],
71     'System Administrator': ["Bachelor", "Master"],
72     'IT Support Specialist': ["High School", "Bachelor"],
73     'Sales Manager': ["Master", "PhD"],
74     'Sales Consultant': ["Bachelor", "Master", "PhD"],
75     'Sales Specialist': ["Bachelor", "Master", "PhD"],
76     'Sales Representative': ["Bachelor"],
77     'Marketing Manager': ["Bachelor", "Master", "PhD"],
78     'SEO Specialist': ["High School", "Bachelor"],
79     'Content Creator': ["High School", "Bachelor"],
80     'Marketing Coordinator': ["Bachelor"],
81     'Finance Manager': ["Master", "PhD"],
82     'Accountant': ["Bachelor"],
83     'Financial Analyst': ["Bachelor", "Master", "PhD"],
84     'Accounts Payable Specialist': ["Bachelor"],
85     'Operations Manager': ["Bachelor", "Master"],
86     'Operations Analyst': ["Bachelor", "Master"],
87     'Logistics Coordinator': ["Bachelor"],
88     'Inventory Specialist': ["High School", "Bachelor"],
89     'Customer Service Manager': ["Bachelor", "Master", "PhD"],
90     'Customer Service Representative': ["High School", "Bachelor"],

```

```

88     'Support Specialist': ["High School", "Bachelor"],
89     'Customer Success Manager': ["Bachelor", "Master", "PhD"],
90     'Help Desk Technician': ["High School", "Bachelor"]
91 }
92
93 # Hiring Date
94 # Define custom probability weights for each year
95 year_weights = {
96     2015: 5, # 15% probability
97     2016: 8, # 15% probability
98     2017: 17, # 20% probability
99     2018: 9, # 15% probability
100    2019: 10, # 10% probability
101    2020: 11, # 10% probability
102    2021: 5, # 8% probability
103    2022: 12, # 5% probability
104    2023: 14, # 2% probability
105    2024: 9 # 2% probability
106 }
107
108 # Generate a random date based on custom probabilities
109 def generate_custom_date(year_weights):
110     year = random.choices(list(year_weights.keys()),
111 weights=list(year_weights.values()))[0]
112     month = random.randint(1, 12)
113     day = random.randint(1, 28) # Assuming all months have 28 days for
simplicity
114     return fake.date_time_between(start_date=datetime(year, 1, 1),
115 end_date=datetime(year, 12, 31))
116
117 def generate_salary(department, job_title):
118     salary_dict = {
119         'HR': {
120             'HR Manager': np.random.randint(60000, 90000),
121             'HR Coordinator': np.random.randint(50000, 60000),
122             'Recruiter': np.random.randint(50000, 70000),
123             'HR Assistant': np.random.randint(50000, 60000)
124         },
125         'IT': {
126             'IT Manager': np.random.randint(80000, 120000),
127             'Software Developer': np.random.randint(70000, 95000),
128             'System Administrator': np.random.randint(60000, 90000),
129             'IT Support Specialist': np.random.randint(50000, 60000)
130         },
131         'Sales': {
132             'Sales Manager': np.random.randint(70000, 110000),
133             'Sales Consultant': np.random.randint(60000, 90000),
134             'Sales Specialist': np.random.randint(50000, 80000),
135             'Sales Representative': np.random.randint(50000, 70000)
136         }
137     }

```

```

135         'Marketing': {
136             'Marketing Manager': np.random.randint(70000, 100000),
137             'SEO Specialist': np.random.randint(50000, 80000),
138             'Content Creator': np.random.randint(50000, 60000),
139             'Marketing Coordinator': np.random.randint(50000, 70000)
140         },
141         'Finance': {
142             'Finance Manager': np.random.randint(80000, 120000),
143             'Accountant': np.random.randint(50000, 80000),
144             'Financial Analyst': np.random.randint(60000, 90000),
145             'Accounts Payable Specialist': np.random.randint(50000, 60000)
146         },
147         'Operations': {
148             'Operations Manager': np.random.randint(70000, 100000),
149             'Operations Analyst': np.random.randint(50000, 80000),
150             'Logistics Coordinator': np.random.randint(50000, 60000),
151             'Inventory Specialist': np.random.randint(50000, 60000)
152         },
153         'Customer Service': {
154             'Customer Service Manager': np.random.randint(60000, 90000),
155             'Customer Service Representative': np.random.randint(50000,
156             60000),
157             'Support Specialist': np.random.randint(50000, 60000),
158             'Help Desk Technician': np.random.randint(50000, 80000)
159         }
160     }
161     return salary_dict[department][job_title]
162
163 # Generate the dataset
164 data = []
165
166 for _ in range(num_records):
167     employee_id = f"00-{random.randint(10000000, 99999999)}"
168     first_name = fake.first_name()
169     last_name = fake.last_name()
170     gender = np.random.choice(['Female', 'Male'], p=[0.46, 0.54])
171     state = np.random.choice(states, p=state_prob)
172     city = np.random.choice(states_cities[state])
173     hiredate = generate_custom_date(year_weights)
174     #termdate
175     department = np.random.choice(departments, p=departments_prob)
176     job_title = np.random.choice(jobtitles[department],
177     p=jobtitles_prob[department])
178     education_level = np.random.choice(education_mapping[job_title])
179     performance_rating = np.random.choice(['Excellent', 'Good', 'Satisfactory',
180     'Needs Improvement'], p=[0.12, 0.5, 0.3, 0.08])
181     overtime = np.random.choice(['Yes', 'No'], p=[0.3, 0.7])
182     salary = generate_salary(department, job_title)
183
184     data.append([
185         employee_id,

```

```
180     first_name,
181     last_name,
182     gender,
183     state,
184     city,
185     hiredate,
186     department,
187     job_title,
188     education_level,
189     salary,
190     performance_rating,
191     overtime
192 ])
193
194 ## Create DataFrame
195 columns = [
196     'employee_id',
197     'first_name',
198     'last_name',
199     'gender',
200     'state',
201     'city',
202     'hiredate',
203     'department',
204     'job_title',
205     'education_level',
206     'salary',
207     'performance_rating',
208     'overtime'
209 ]
210
211 df = pd.DataFrame(data, columns=columns)
212
213 # Add Birthdate
214 def generate_birthdate(row):
215     age_distribution = {
216         'under_25': 0.11,
217         '25_34': 0.25,
218         '35_44': 0.31,
219         '45_54': 0.24,
220         'over_55': 0.09
221     }
222     age_groups = list(age_distribution.keys())
223     age_probs = list(age_distribution.values())
224     age_group = np.random.choice(age_groups, p=age_probs)
225
226     if any('Manager' in title for title in row['job_title']):
227         age = np.random.randint(30, 65)
228     elif row['education_level'] == 'PhD':
229         age = np.random.randint(27, 65)
```

```

227     elif age_group == 'under_25':
228         age = np.random.randint(20, 25)
229     elif age_group == '25_34':
230         age = np.random.randint(25, 35)
231     elif age_group == '35_44':
232         age = np.random.randint(35, 45)
233     elif age_group == '45_54':
234         age = np.random.randint(45, 55)
235     else:
236         age = np.random.randint(56, 65)
237
238     birthdate = fake.date_of_birth(minimum_age=age, maximum_age=age)
239     return birthdate
240
241 # Apply the function to generate birthdates
242 df['birthdate'] = df.apply(generate_birthdate, axis=1)
243
244 # Terminations
245 # Define termination distribution
246 year_weights = {
247     2015: 5,
248     2016: 7,
249     2017: 10,
250     2018: 12,
251     2019: 9,
252     2020: 10,
253     2021: 20,
254     2022: 10,
255     2023: 7,
256     2024: 10
257 }
258
259 # Calculate the total number of terminated employees
260 total_employees = num_records
261 termination_percentage = 0.112 # 11.2%
262 total_terminated = int(total_employees * termination_percentage)
263
264 # Generate termination dates based on distribution
265 termination_dates = []
266 for year, weight in year_weights.items():
267     num_terminations = int(total_terminated * (weight / 100))
268     termination_dates.extend([year] * num_terminations)
269
270 # Randomly shuffle the termination dates
271 random.shuffle(termination_dates)
272
273 # Assign termination dates to terminated employees
274 terminated_indices = df.index[:total_terminated]
275 for i, year in enumerate(termination_dates[:total_terminated]):
276     df.at[terminated_indices[i], 'termdate'] = datetime(year, 1, 1) +
277     timedelta(days=random.randint(0, 365))

```

```

273
274
275 # Assign None to termdate for employees who are not terminated
276 df['termdate'] = df['termdate'].where(df['termdate'].notnull(), None)
277
278 # Ensure termdate is at least 6 months after hiredate
279 df['termdate'] = df.apply(lambda row: row['hiredate'] + timedelta(days=180) if
280 row['termdate'] and row['termdate'] < row['hiredate'] + timedelta(days=180) else
281 row['termdate'], axis=1)
282
283 education_multiplier = {
284     'High School': {'Male': 1.03, 'Female': 1.0},
285     'Bachelor': {'Male': 1.115, 'Female': 1.0},
286     'Master': {'Male': 1.0, 'Female': 1.07},
287     'PhD': {'Male': 1.0, 'Female': 1.17}
288 }
289
290 # Function to calculate age from birthdate
291 def calculate_age(birthdate):
292     today = pd.Timestamp('today')
293     age = today.year - birthdate.year - ((today.month, today.day) <
294 (birthdate.month, birthdate.day))
295     return age
296
297 # Function to calculate the adjusted salary
298 def calculate_adjusted_salary(row):
299     base_salary = row['salary']
300     gender = row['gender']
301     education = row['education_level']
302     age = calculate_age(row['birthdate'])
303
304     # Apply education multiplier
305     multiplier = education_multiplier.get(education, {}).get(gender, 1.0)
306     adjusted_salary = base_salary * multiplier
307
308     # Apply age increment (between 0.1% and 0.3% per year of age)
309     age_increment = 1 + np.random.uniform(0.001, 0.003) * age
310     adjusted_salary *= age_increment
311
312     # Ensure the adjusted salary is not lower than the base salary
313     adjusted_salary = max(adjusted_salary, base_salary)
314
315     # Round the adjusted salary to the nearest integer
316     return round(adjusted_salary)
317
318 # Apply the function to the DataFrame
319 df['salary'] = df.apply(calculate_adjusted_salary, axis=1)
320
321 # Convert 'hiredate' and 'birthdate' to datetime
322 df['hiredate'] = pd.to_datetime(df['hiredate']).dt.date

```

```
319 df['birthdate'] = pd.to_datetime(df['birthdate']).dt.date
320 df['termdate'] = pd.to_datetime(df['termdate']).dt.date
321
322 print(df)
323
324 # Save to CSV
325 df.to_csv('HumanResources.csv', index=False)
```

Copyright © 2025 DATA with BARAA | Powered by DATA with BARAA