HUMAN RESOURCES ANALYTICS

وزارة الاتصالات وتكنولوجيا المعلومات
Ministry of Communications and Information Technology

Digital Egypt Pioneers

Under supervision of Eng. ZyadAyman

Project’s members:

-Ali tarek elgenedy (leader)

-Habiba walid aly

-Ibrahim wael Ismael

-Ahmed saber Abdelshafy

Group’s name: DEPI_GIZ2_DAT1_S3

REPORT

- In any organization, managing and analyzing employee data is crucial for efficient human resource management, strategic decision-making, and operational effectiveness. Employee data encompasses a wide range of personal, demographic, and professional attributes that provide valuable insights into workforce composition, employment history, and trends. The dataset presented here includes extensive details about employees, capturing critical information such as identification details, personal demographics, employment records, and geographic locations. This introduction provides an in-depth overview of the various attributes included in the dataset and their significance in workforce analysis.

- Below is a breakdown of each column:

Emp ID: Unique identifier for each employee.
Example: 857211, 514341, etc.
Type: Numeric (Integer).

Jop Title: Title or prefix of the employee's name (e.g., Eng, Data analyst).
Example: (e.g., Eng, Data analyst).
Type: Categorical (String).

First Name: The first name of the employee.
Example: Hermila, Antonio, Sebastian.
Type: String.

Middle Initial: refers to the first letter of an employee's middle name (if they have one). For example, if someone's full name is **John Michael Smith**, their middle initial would be **M** (the first letter of the middle name "Michael").
Example: J, Q, S.
Type: String (Single character).

Last Name: The last name of the employee.
Example: Suhr, Joy, Moores.
Type: String.

Gender: Gender of the employee.
Example: F(Female), M (Male).
Type: Categorical (String).

E Mail: Email address of the employee.
Example: hermila.suhr@gmail.com, antonio.joy@yahoo.com.
Type: String.

Father's Name: Full name of the employee's father.
Example: Todd Suhr, Clark Joy.
Type: String.

Mother's Name: Full name of the employee's mother.
Example: Cathrine Suhr, Clarisa Joy.
Type: String.

Mother's Maiden Name: Maiden name of the employee's mother (last name before marriage).
Example: Hinojosa, Gagliardi.
Type: String.

Date of Birth: Employee's date of birth in `MM/DD/YYYY` format.
Example: 9/4/1992, 12/24/1989.
Type: Date.

Time of Birth: Time of birth of the employee in HH:MM:SS AM/PM format.
Example: 4:29:56 AM, 8:01:44 AM.
Type: Time.

Age in Yrs: Age of the employee in years (calculated as of a specific date).If an employee was born on **September 4, 1992**, and today's date is **February 20, 2025**, the employee would be **24.91 years old** (approximately 24 years and 11 months old).
Example: 24.91, 27.61.
Type: Numeric (Float).

Weight in Kgs: Weight of the employee in kilograms.
Example: 57, 55.
Type: Numeric (Integer).

Date of Joining: Date when the employee joined the company in MM/DD/YYYY format.

Example: 9/9/2014, 8/2/2011.
Type: Date.

Quarter of Joining: Quarter of the year when the employee joined (Q1, Q2, Q3, Q4).
Example: Q3, Q2.
Type: Categorical (String).

Half of Joining: Half of the year when the employee joined (H1 for Jan-Jun, H2 for Jul-Dec).
Example: H2, H1.
Type: Categorical (String).

Year of Joining: Year when the employee joined the company.
Example: 2014, 2011.
Type: Numeric (Integer).

Month of Joining: Month number when the employee joined (1-12).
Example: 9, 8.
Type: Numeric (Integer).

Month Name of Joining: Full name of the month when the employee joined.
Example: September, August.
Type: Categorical (String).

Short Month: Abbreviated name of the month when the employee joined.
Example: Sep, Aug.
Type: Categorical (String).

Day of Joining: Day of the month when the employee joined.
Example: 9, 2.
Type: Numeric (Integer).

DOW of Joining: Day of the week when the employee joined (full name).
Example: Tuesday, Sunday.
Type: Categorical (String).

Short DOW: Abbreviated day of the week when the employee joined.
Example: Tue, Sun.

Type: Categorical (String).

Age in Company (Years): Number of years the employee has been with the company (as of a specific date).
Example: 2.88, 5.99.
Type: Numeric (Float).

Salary: Current salary of the employee.
Example: 168991, 53504.
Type: Numeric (Integer).

Last % Hike: Percentage of the last salary hike received by the employee.
Example: 12%, 30%.
Type: String (Percentage).

SSN: Social Security Number of the employee (hyphen-separated).
Example: 275-17-8844, 646-23-6213.
Type: String.

Phone No: Phone number of the employee (hyphen-separated).
Example: 479-539-4593, 229-234-6154.
Type: String.

Place Name: Name of the place where the employee resides.
Example: Peach Orchard, Rocky Ford.
Type: String.

County: County where the employee resides.
Example: Clay, Screven.
Type: String.

City: City where the employee resides.
Example: Peach Orchard, Rocky Ford.
Type: String.

State: state where the employee resides (abbreviated).
Example: AR, GA.

Type: Categorical (String).

Zip: ZIP code of the employee's residence.
Example: 72453, 30455.
Type: Numeric (Integer).

Region: Geographic region of the employee's residence (e.g., South, Northeast).
Example: South, Northeast.
Type: Categorical (String).

User Name: Username of the employee (likely for company systems).
Example: hjsuhr, aqjoy.
Type: String.

Password: Password of the employee (likely for company systems).
Example: oZ%{<6wN!A, 7_[%FE;saZ:B.
Type: String

➢ Data cleaning and preprocessing

In this project, we will be working with a dataset using **Python** and the **Pandas** library for data analysis and manipulation. The following sections will outline the essential steps in loading, exploring, and preparing the dataset. Each step includes Python code and an explanation of its functionality.

This point will cover:

- Importing necessary libraries
- Loading and inspecting the dataset
- Handling missing values and duplicate entries
- Extracting key information from the dataset

By following these steps, we aim to ensure that the data is clean, structured, and ready for further analysis.

1. **Importing and loading**
   This line imports the pandas library and gives it an alias (pd). also loads the dataset from a 'CSV' (Comma Separated Values) file located at the specified location.

```
import pandas as pd  # to read data
import matplotlib.pyplot as plt
import seaborn as sns
# Read the CSV file
df = pd.read_csv(r"D:\DATA ANALYSIS\DEPI\Final project Depi\Dataset\500000 Records - Copy.csv")
```

2. Checking the shape of the DataFrame

The data.shape command returns the ‘dimensions’ of the dataset, specifically the number of rows and columns. This is useful for quickly understanding the size of the dataset.

```
print(data.shape)  # check How many rows and columns do we have
```

```
Help  Variable Explorer  Plots  Files  Find
Console 1/A
In [22]: runfile('C:/Users/USER/untitled0.py', wdir='C:/Users/USER')
(505743, 37)

In [23]:
```

3. Counting the number of missing values in each column

The data.isnull() method creates a boolean DataFrame where each value is True if the value in that position is missing (null), and False if the value is present. The .sum() function adds up the number of True values for each column, effectively counting how many ‘missing’ values exist in each column. This helps identify columns with missing data that may need to be cleaned or imputed.

```
########## Handling Nulls ##########

# Check if there is is any null value
print(df.isnull().sum())
```

```
Console 1/A
In [25]: runfile('C:/Users/USER/untitled0.py', wdir='C:/Users/USER')
Name Prefix                0
First Name                 0
Middle Initial             0
Last Name                  0
Gender                     0
E Mail                  3625
Father's Name              0
Mother's Name              0
Mother's Maiden Name       0
Date of Birth              0
Time of Birth              0
Age in Yrs.                0
Weight in Kgs.             0
Date of Joining            0
Quarter of Joining         0
Half of Joining            0
Year of Joining            0
Month of Joining           0
Month Name of Joining      0
Short Month                0
Day of Joining             0
DOW of Joining             0
Short DOW                  0
Age in Company (Years)     0
```

```
Console 1/A
Day of Joining             0
DOW of Joining             0
Short DOW                  0
Age in Company (Years)     0
Salary                     0
Last % Hike                0
SSN                        0
Phone No.               1974
Place Name                 0
County                     0
City                    1688
State                      0
Zip                        0
Region                  1369
User Name                  0
Password                   0
EmployeeID                 0
dtype: int64

In [26]:
```

4. To fill the missing values

The .fillna() method is used to replace missing values in the dataset. Missing email, city, and region values are replaced with 'Unknown' to maintain data consistency. For the phone number column, missing values are filled with the most frequent (mode) phone number in the dataset to ensure data integrity. This step helps handle missing data without losing valuable records.

```
#To fill Missing Values

df['E Mail'] = df['E Mail'].fillna('Unknown')
df['Phone No. '] = df['Phone No. '].fillna(df['Phone No. '].mode()[0])
df['City'] = df['City'].fillna('Unknown')
df['Region'] = df['Region'].fillna('Unknown')
```

5. **Checking for Null Values After Filling**

The df.isnull().sum() method is used to check for any remaining missing values in the dataset after filling them. It returns the count of null values for each column. If all values are 0, it confirms that missing data has been successfully handled. This step ensures data completeness before further analysis.

```
Check if there is is any null value after Filling
int(df.isnull().sum())
```

```
Place Name              0
County                  0
City                    0
State                   0
Zip                     0
Region                  0
User Name               0
Password                0
EmployeeID              0
dtype: int64
```

```
Console 1/A

In [33]: runfile('C:/Users/USER/untitled0.py', wdir='C:/Users/USER')
Name Prefix             0
First Name              0
Middle Initial          0
Last Name               0
Gender                  0
E Mail                  0
Father's Name           0
Mother's Name           0
Mother's Maiden Name    0
Date of Birth           0
Time of Birth           0
Age in Yrs.             0
Weight in Kgs.          0
Date of Joining         0
Quarter of Joining      0
Half of Joining         0
Year of Joining         0
Month of Joining        0
Month Name of Joining   0
Short Month             0
Day of Joining          0
DOW of Joining          0
Short DOW               0
Age in Company (Years)  0
Salary                  0
Last % Hike             0
SSN                     0
Phone No.               0
```

6. **Checking for Duplicate Values**

The df.duplicated().sum() method checks for duplicate rows in the dataset. It returns the total count of duplicate rows, helping to identify redundant entries that may affect data integrity. In this case, the output shows 5743 duplicate rows, indicating that further action, such as removal or handling of duplicates, may be required.

```
# Check if there any duplicated value
print(df.duplicated().sum())
```

```
In [34]: runfile('C:/Users/USER/untitled0.py', wdir='C:/Users/USER')
5743

In [35]:
```

7. **Dropping Duplicate Rows**

The df.drop_duplicates(inplace=True) method removes duplicate rows from the dataset. The inplace=True parameter ensures that the changes are applied directly to the DataFrame without needing to reassign it. This step is crucial for maintaining data integrity and preventing redundant entries from skewing analysis.

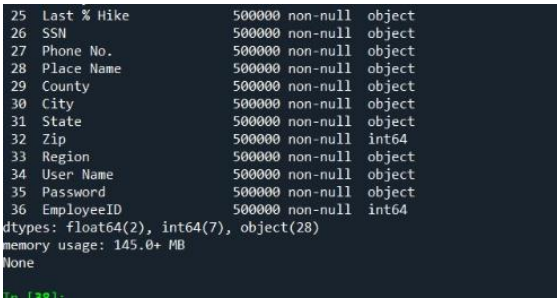
```
#To Drop Duplicates
df.drop_duplicates(inplace=True)
```

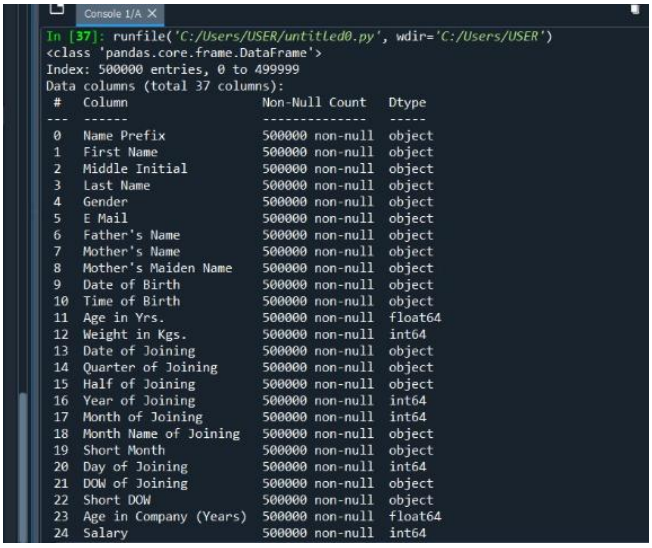
8. Verifying Duplicate Removal in the Dataset

After removing duplicate rows, the df.duplicated().sum() method is used to check if any duplicates remain in the DataFrame. This function returns the count of duplicate rows, ensuring data consistency. The output 0 in the console confirms that all duplicate entries have been successfully removed, and the dataset is now free from redundancies. This step is crucial in data preprocessing to maintain accuracy and reliability in further analysis.

9. Getting basic information about the dataset

The data.info() method provides a 'summary' of the DataFrame. It includes: - The number of entries (rows). - The column names. - The number of non-null (non-missing) entries in each column. - The data type of each column (e.g., integer, float, string).





10. Statistical Overview of the Dataset

To gain a better understanding of the dataset, the df.describe() function was utilized. This function provides a comprehensive statistical summary of all numerical columns, offering key insights into the dataset's distribution and characteristics.

The output includes:

Count: The total number of non-null values for each variable, ensuring data completeness.

Mean: The average value of each numerical attribute, giving an estimate of central tendency.

Standard Deviation (std): A measure of data dispersion, indicating how spread out the values are.

Minimum (min) and Maximum (max): The smallest and largest values recorded in each column.

Quartiles (25%, 50%, 75%): These percentiles provide insights into the distribution and median values of the dataset.

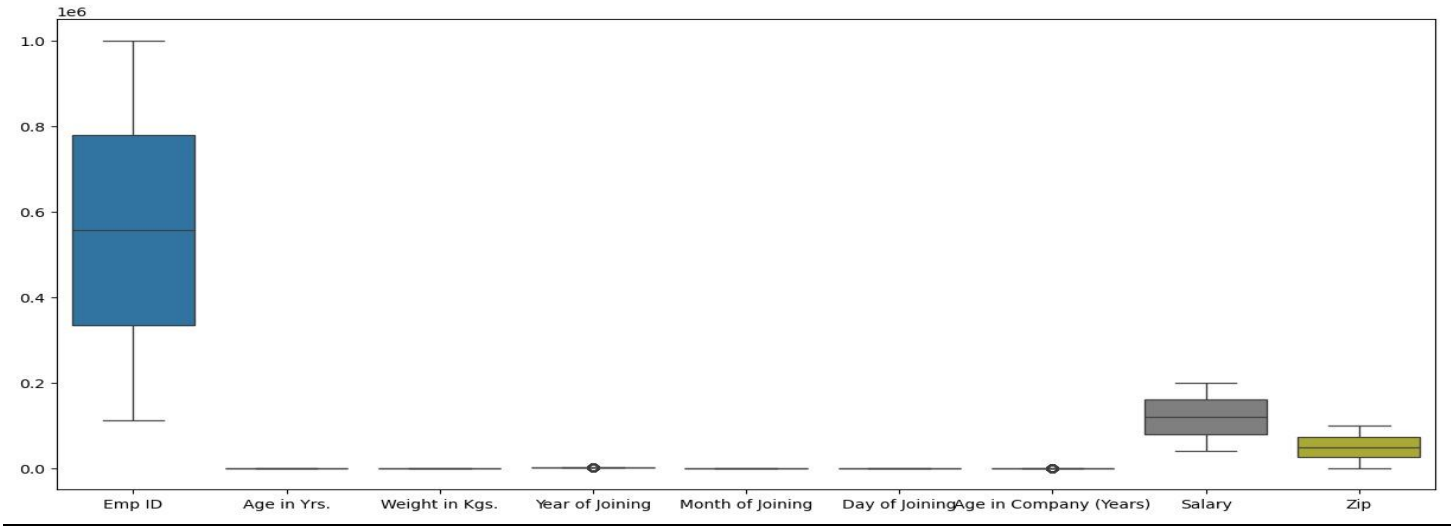
This statistical summary is crucial in the data preprocessing stage, allowing for the detection of potential anomalies, missing values, and overall data patterns. Understanding these metrics helps ensure the dataset's accuracy and reliability before proceeding with further analysis or model development**.**

```
# Some statistical description
print(df.describe())
```

```
In [38]: runfile('C:/Users/USER/untitled0.py', wdir='C:/Users/USER')
       Age in Yrs.  Weight in Kgs.  ...            Zip     EmployeeID
count  500000.000000   500000.000000  ...  500000.000000  500000.000000
mean       40.480371       59.991598  ...   49598.599586  250000.500000
std        11.244833       13.722146  ...   27763.227710  144337.711634
min        21.000000       40.000000  ...     210.000000       1.000000
25%        30.740000       50.000000  ...   26219.000000  125000.750000
50%        40.490000       57.000000  ...   48701.000000  250000.500000
75%        50.200000       70.000000  ...   72860.000000  375000.250000
max        60.000000       90.000000  ...   99950.000000  500000.000000

[8 rows x 9 columns]

In [39]:
```

11.Check outliers and saved cleaned data

```
#check outliers

# We check the outliers of every feature using boxplot
plt.figure(figsize=(15,7))
sns.boxplot(data=df)
plt.show()

#there is no outliers

# Save cleaned data
df.to_csv(r'D:\DATA ANALYSIS\DEPI\Final project\500000 Records.csv\500000 Records.csv', index=False)S
```



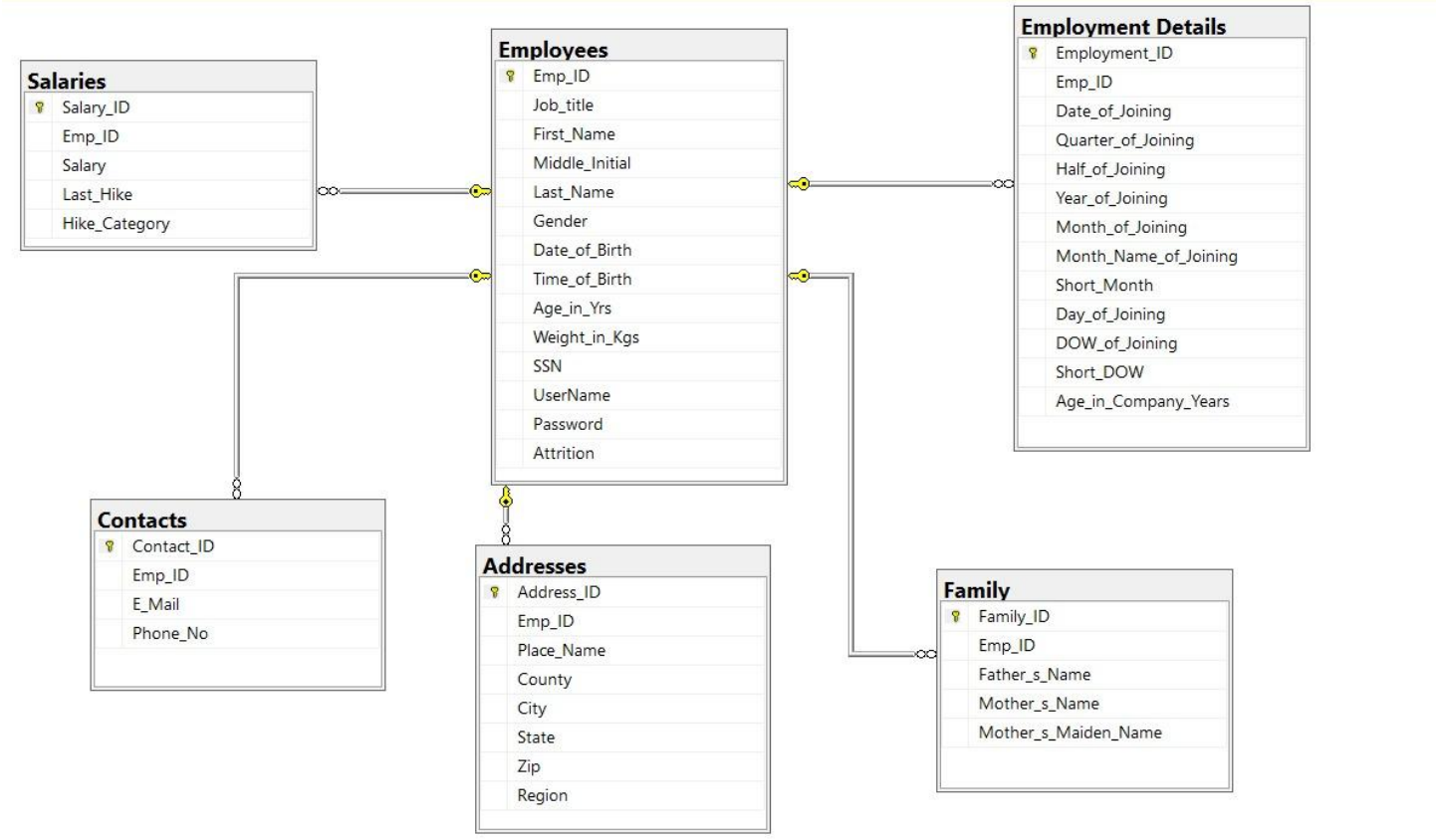
➢ Build Data Module:

Previously, we examined the **original structure** of the dataset, which contained redundant and unstructured data. To improve **data integrity, eliminate redundancy, and optimize performance**, we performed **normalization using SQL, and then we applied a star schema for fast querying and reporting.**

Normalization is a fundamental concept in relational database design. It refers to the process of organizing data in a database to reduce redundancy and improve data integrity. The goal is to ensure that each piece of information is stored only once and in the most appropriate place.

When data is not normalized, the same information may be repeated in multiple places. This can lead to inconsistencies, larger storage requirements, and difficulty in updating data. Normalization solves this by dividing large tables into smaller, more manageable ones and defining relationships between them.

Below is the **normalized database model** after applying **SQL normalization techniques and then Star schema design:**



How It Works

the database is designed around the central **Employees** table, which stores key personal data about each employee like name, age, gender, and job title.

Instead of putting everything in one big table, related data is stored in separate, connected tables:

- **Salaries**: Contains salary information. Linked by `Emp_ID`, this ensures salary data is separate from personal details.
- **Contacts**: Holds employee emails and phone numbers, also linked by `Emp_ID`.
- **Addresses**: Stores address information separately. This makes it easier to manage if an employee changes address.
- **Employment Details**: Contains details about when and how the employee joined the company.
- **Family**: Stores family-related data like parents' names.

This design uses **foreign keys** to link the related tables through the common field `Emp_ID`. For example, one employee can have one salary record, one contact record, one address, etc.

Benefits of Normalization

1. **Eliminates Redundancy**: No repeated information in multiple places.
2. **Improves Consistency**: Updating a single piece of data doesn't require multiple changes.
3. **Makes Maintenance Easier**: Tables are simpler and easier to manage.
4. **Enhances Query Performance**: Smaller tables are faster to search and process.

Key improvements after applying star schema design:

- **Simple & Easy to Understand –** The design is intuitive and easy to query**.**
- **Faster Query Performance** – Uses **denormalized** tables, reducing complex joins.
- **Efficient for OLAP & Reporting –** Ideal for Business Intelligence (BI) tools like Power BI & Tableau.
- **Better Aggregation –** Designed for quick summarization of data**.**

➢ Analysis Questions Phase:

Before we start analyzing data and asking business questions**,** we applied the exploratory data analysis (**EDA**) step, which helps to understand the dataset's structure, detect patterns, and identify any issues. The data has been prepared for this phase.

We conducted an initial study of the Human Resources (HR) business to define the strategic questions and Key Performance Indicators (KPI's), as well as to determine the objectives we will focus on in the coming period.

The study helped us understand the overall context of HR data, enabling us to prioritize and direct the analysis in alignment with business goals**.**

1. Hiring & Workforce Distribution

- Q1: Hiring Trend Over Time (Year – Quarter – Month – Weekday)
- Q4: Total New Hires (This Year)
- Q5: Percentage Increase/Decrease in Hiring (YOY)
- Q2: Number of Employees in Each State
- Q17: Top & Bottom 5 States by Number of Employees
- Q18: Top 5 States by Percentage of Total Employees
- Q21: Number of Employees by Gender in Each Region
- Q8: Hiring Trends by Gender Over Time

2. Employee Demographics

- Q3: Average Age at Hiring
- Q6: Gender Ratio (Male vs Female)
- Q9: Average Age by Region
- Q10: Number of Employees by Age Group
- Q19: Average Age by Region
- Q25: Average Age in Company by Gender
- Q26: Top & Bottom 5 States by Average Age in Company
- Q22: Average Age in Company by Region
- Q28: Percentage of Employees with Same Last Name
- Q29: Name Prefix Analysis (Count, Average Salary, Average Hike)
- Q33: Email Classification (Individual vs Company)
- Q40: Average Age (Left vs Stayed)

3. Experience & Compensation

- Q7: Average Salary by Gender
- Q11: Correlation Between Salary and Age in Company
- Q12: Average Salary by Region
- Q13: Number of Employees by Last % Hike Category
- Q14: Number of Males and Females by Last % Hike Category
- Q15: Distribution by Age in Company (Experience)
- Q16: Salary Box Plot Distribution and Outliers
- Q20: Average Salary in Each State
- Q23: Average % Hike by Age Group
- Q24: Average Salary by Age Group
- Q27: Correlation Matrix (Salary, Age in Company, % Hike)
- Q31: Distribution by Career Stage
- Q32: Salary Comparison Across Career Stages

4. Success Factors of Top 10% High-Earning Employees

- Q30.1: What is the average age of the top 10% employees based on salary
- Q30.2: What is the gender distribution among the top 10% employees
- Q30.3: What is the regional distribution of the top 10% employees
- Q30.4: What is the average years of experience for the top 10% employees
- Q30.5: How does the joining time (quarter/half/year) relate to the salary growth of the top 10% employees

5. Attrition & Exit Analysis

- Q34: Overall Employee Attrition Rate
- Q35: Region with Highest Attrition
- Q36: State with Highest Attrition
- Q37: Average Salary Comparison (Left vs Stayed)
- Q38: Average Tenure Comparison (Left vs Stayed)
- Q39: Attrition Trend Over Time (Year, Quarter, Month)
- Q41: Gender Distribution of Employees Who Left
- Q42: Average Last % Hike (Left vs Stayed)

In our analysis, we adopted a diverse approach to answering the questions by leveraging multiple data analysis tools, each selected based on the nature and complexity of the questions. We primarily used SQL to extract, manipulate, and analyze the data, creating specific views from Category 4 Success Employees to gain deeper insights. These SQL queries allowed us to efficiently structure and filter the data in preparation for further analysis.

Following the SQL-based exploration, we utilized Power BI along with DAX to visualize the data and perform advanced analytics. Power BI enabled us to present interactive dashboards and uncover hidden trends, offering a more comprehensive view of the data. This multi-tool strategy ensured that each question was addressed with the most appropriate and powerful analytical method.

In the sections that follow, we will first walk through the insights derived using SQL, followed by those achieved through Power BI.

First SQL:

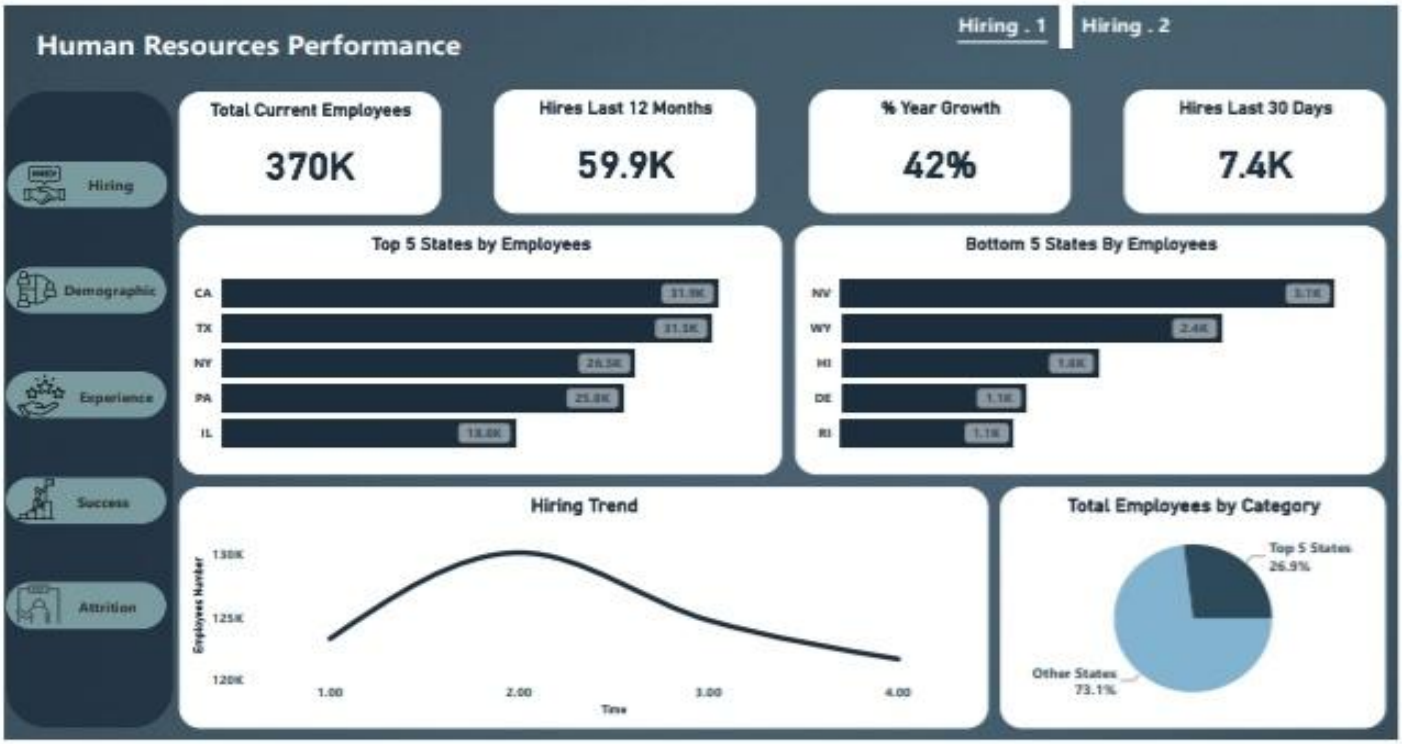
```
------------4. Success Factors of Top 10% High-Earning Employees------------

----Q30.1: What is the average age of the top 10% employees based on salary

ALTER VIEW average_age_of_the_top_10_employees AS

WITH filtering AS (

    SELECT *, NTILE(10) OVER (ORDER BY s.Salary DESC) AS groups

    FROM Salaries s)
```

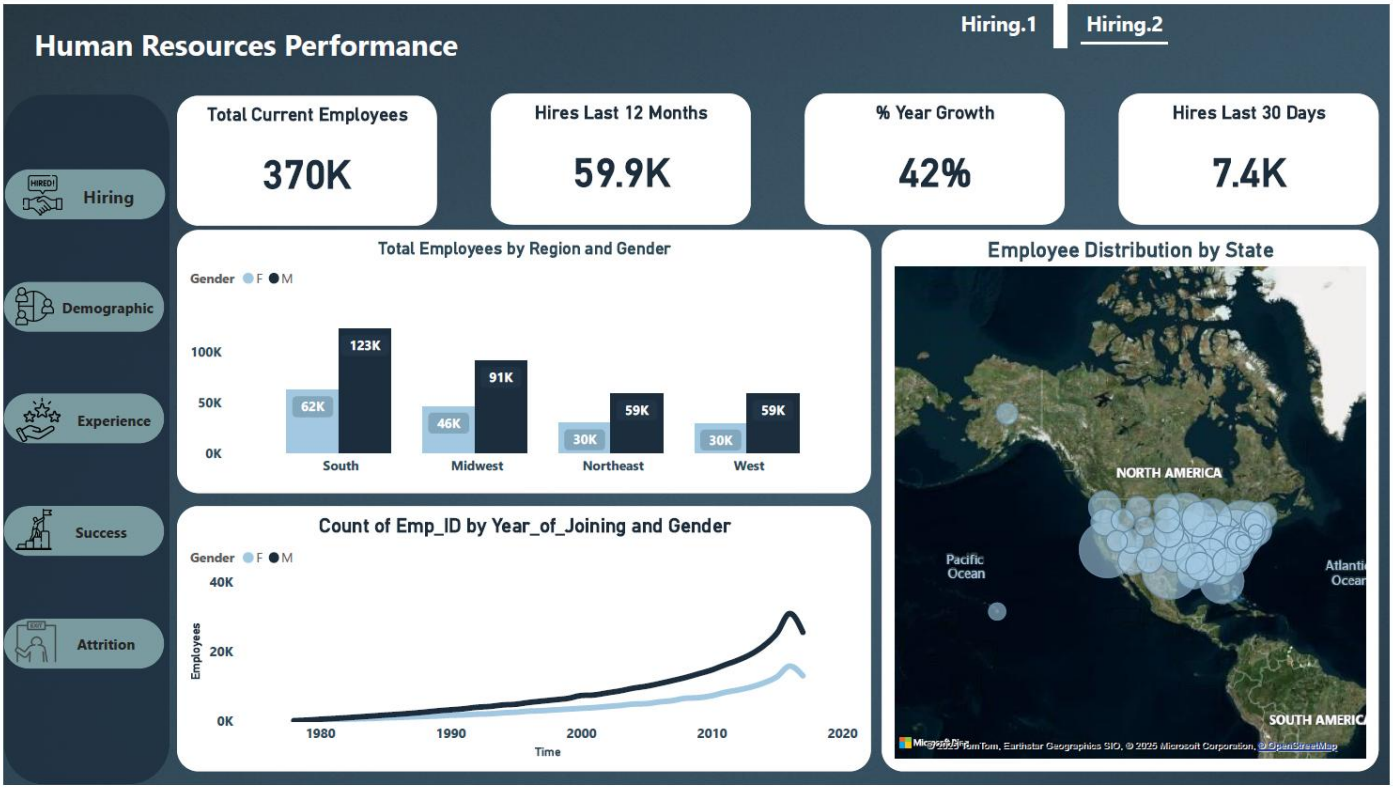
```
SELECT AVG(E.Age_in_Yrs) AS [average age of the top 10 % employees]
FROM filtering f, Employees E
WHERE f.Emp_ID = E.Emp_ID AND f.groups = 1;
----Q30.2: What is the gender distribution among the top 10 % employees based salary
ALTER VIEW the_Gender_distribution_of_the_top_10_employees AS
WITH filtering AS (
    SELECT *, NTILE(10) OVER (ORDER BY s.Salary DESC) AS groups
    FROM Salaries s)
SELECT E.Gender, COUNT(E.Gender) AS [Number of Employees]
FROM Employees E, filtering f
WHERE E.Emp_ID = f.Emp_ID AND f.groups = 1
GROUP BY E.Gender;
----Q30.3: What is the regional distribution of the top 10 % employees based salary
ALTER VIEW the_regional_distribution_of_the_top_10_employees AS
WITH filtering AS (
    SELECT *, NTILE(10) OVER (ORDER BY s.Salary DESC) AS groups
    FROM Salaries s)
SELECT A.Region, COUNT(A.Region) AS [Number of Employees]
FROM Addresses A, filtering f
WHERE A.Emp_ID = f.Emp_ID AND f.groups = 1
GROUP BY A.Region;
----Q30.4: What is the average years of experience for the top 10% employees based salary
ALTER VIEW average_years_of_experience_or_the_top_10_employees AS
WITH filtering AS (
    SELECT *, NTILE(10) OVER (ORDER BY s.Salary DESC) AS groups
    FROM Salaries s)
SELECT AVG(E.Age_in_Company_Years) AS [average years of experience]
FROM [Employment Details] E, filtering f
```

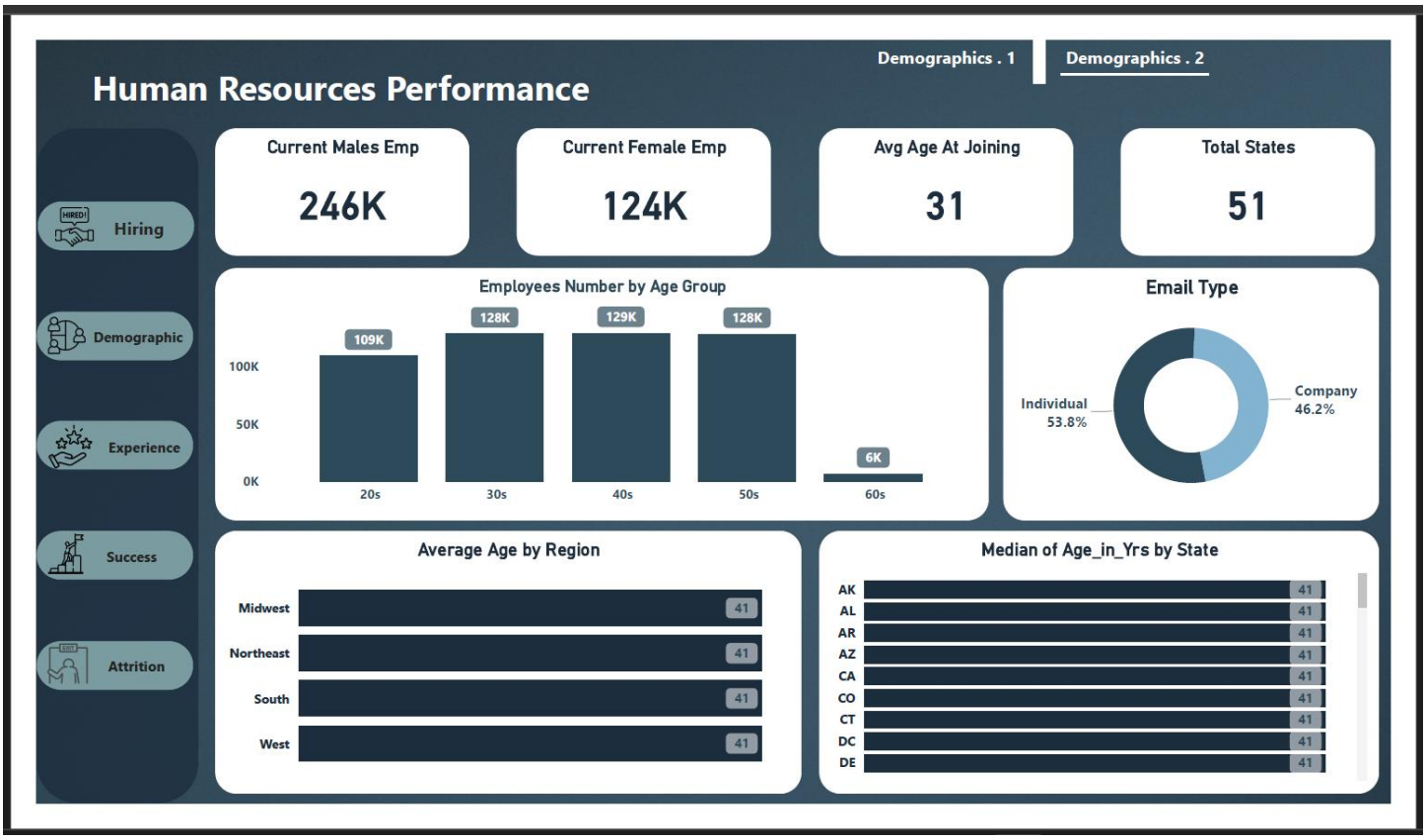
```
WHERE E.Emp_ID = f.Emp_ID AND f.groups = 1;

----Q30.5: How does the joining time (quarter/half/year) relate to the salary growth of the top 10% employees

ALTER VIEW joining_time_top_10_employees AS

WITH filtering AS (

    SELECT *, NTILE(10) OVER (ORDER BY s.Salary DESC) AS groups

    FROM Salaries s)

SELECT

    E.Date_of_Joining,

    E.Year_of_Joining,

    E.Quarter_of_Joining,

    E.Half_of_Joining,

    COUNT(E.Emp_ID) AS Total_Employees

FROM [Employment Details] E

JOIN filtering f ON E.Emp_ID = f.Emp_ID

WHERE f.groups = 1

GROUP BY

    E.Date_of_Joining,

    E.Year_of_Joining,

    E.Quarter_of_Joining,

    E.Half_of_Joining;

----Q99: Attrition status of top 10% employees

ALTER VIEW top_10_percent_attrition_status AS

WITH top_earners AS (

    SELECT *, NTILE(10) OVER (ORDER BY Salary DESC) AS grp

    FROM Salaries)

SELECT

    E.Attrition,

    COUNT(*) AS count_employees,
```

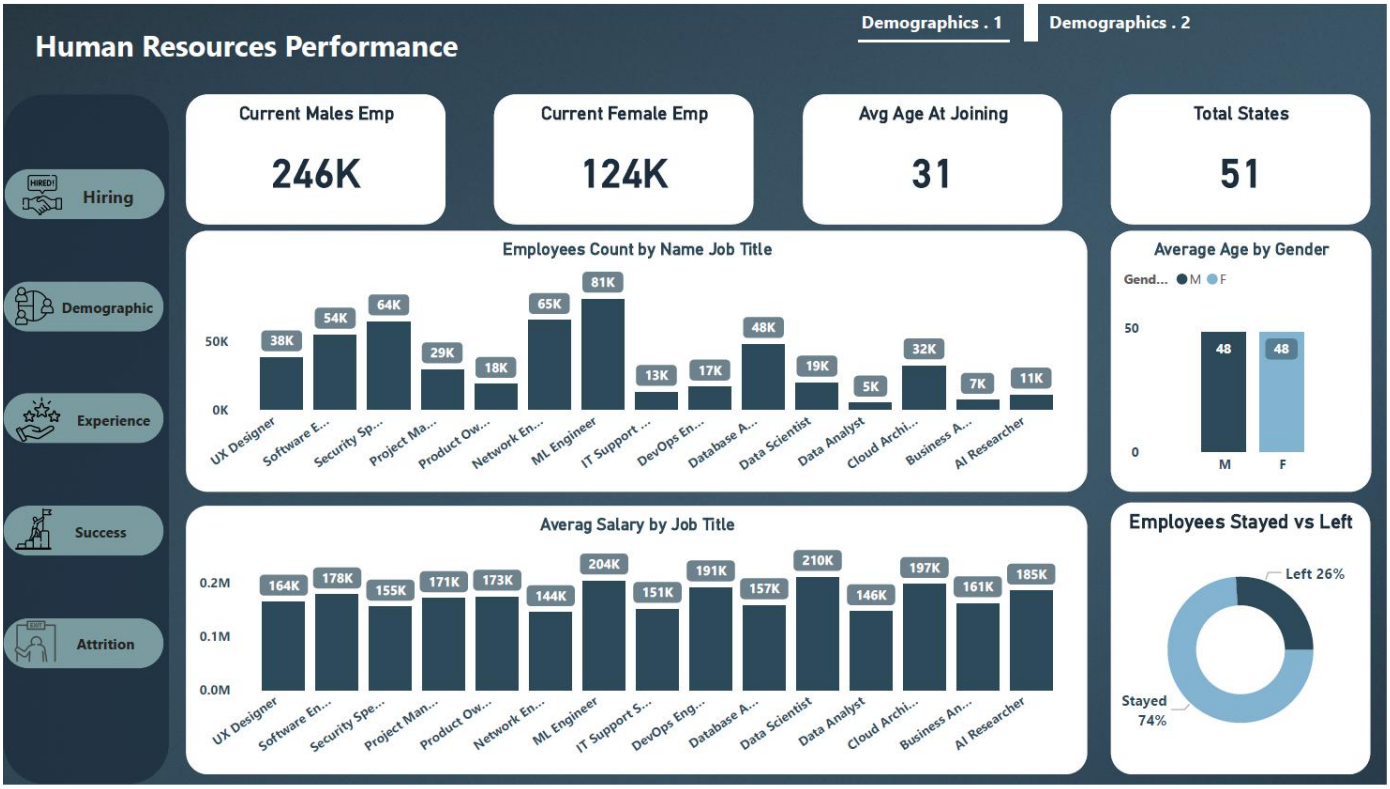
```
    ROUND(COUNT(*) * 1.0 / SUM(COUNT(*)) OVER (), 2) AS percentage_of_top_10
FROM top_earners T
JOIN Employees E ON T.Emp_ID = E.Emp_ID
WHERE T.grp = 1
GROUP BY E.Attrition;
----Q99: Gender distribution of top 10%
ALTER VIEW gender_distribution_top_10_employees AS
WITH top_earners AS (
    SELECT *, NTILE(10) OVER (ORDER BY Salary DESC) AS grp
    FROM Salaries)
SELECT
    E.Gender,
    COUNT(*) AS count_employees,
    ROUND(COUNT(*) * 1.0 / SUM(COUNT(*)) OVER (), 2) AS percentage
FROM top_earners T
JOIN Employees E ON T.Emp_ID = E.Emp_ID
WHERE T.grp = 1
GROUP BY E.Gender;
----Q99: States of top 10% earners
ALTER VIEW top_states_top_10_employees AS
WITH top_earners AS (
    SELECT *, NTILE(10) OVER (ORDER BY Salary DESC) AS grp
    FROM Salaries)
SELECT A.State, COUNT(*) AS count_employees
FROM top_earners T
JOIN Addresses A ON T.Emp_ID = A.Emp_ID
WHERE T.grp = 1
GROUP BY A.State;
```

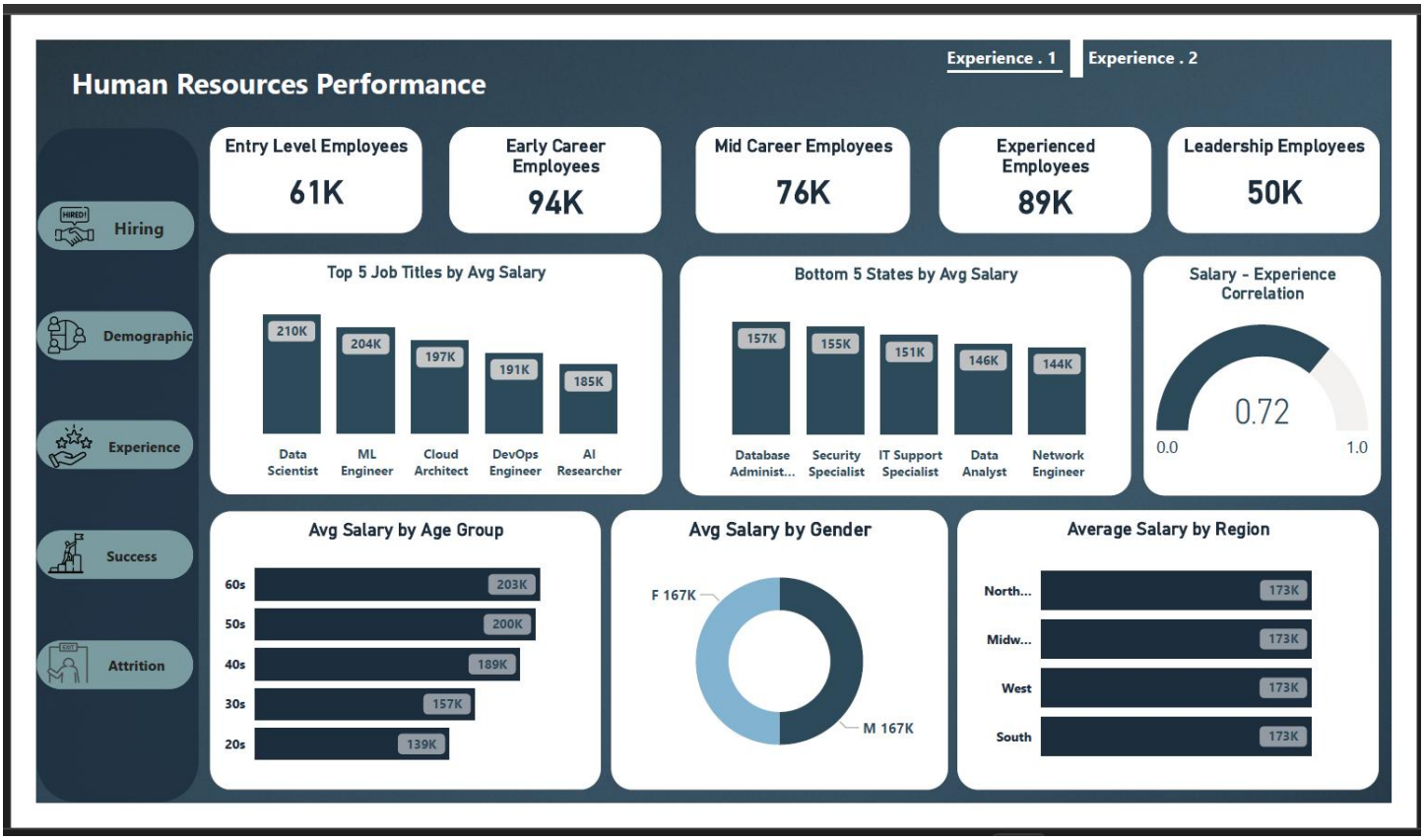
Second Power BI:

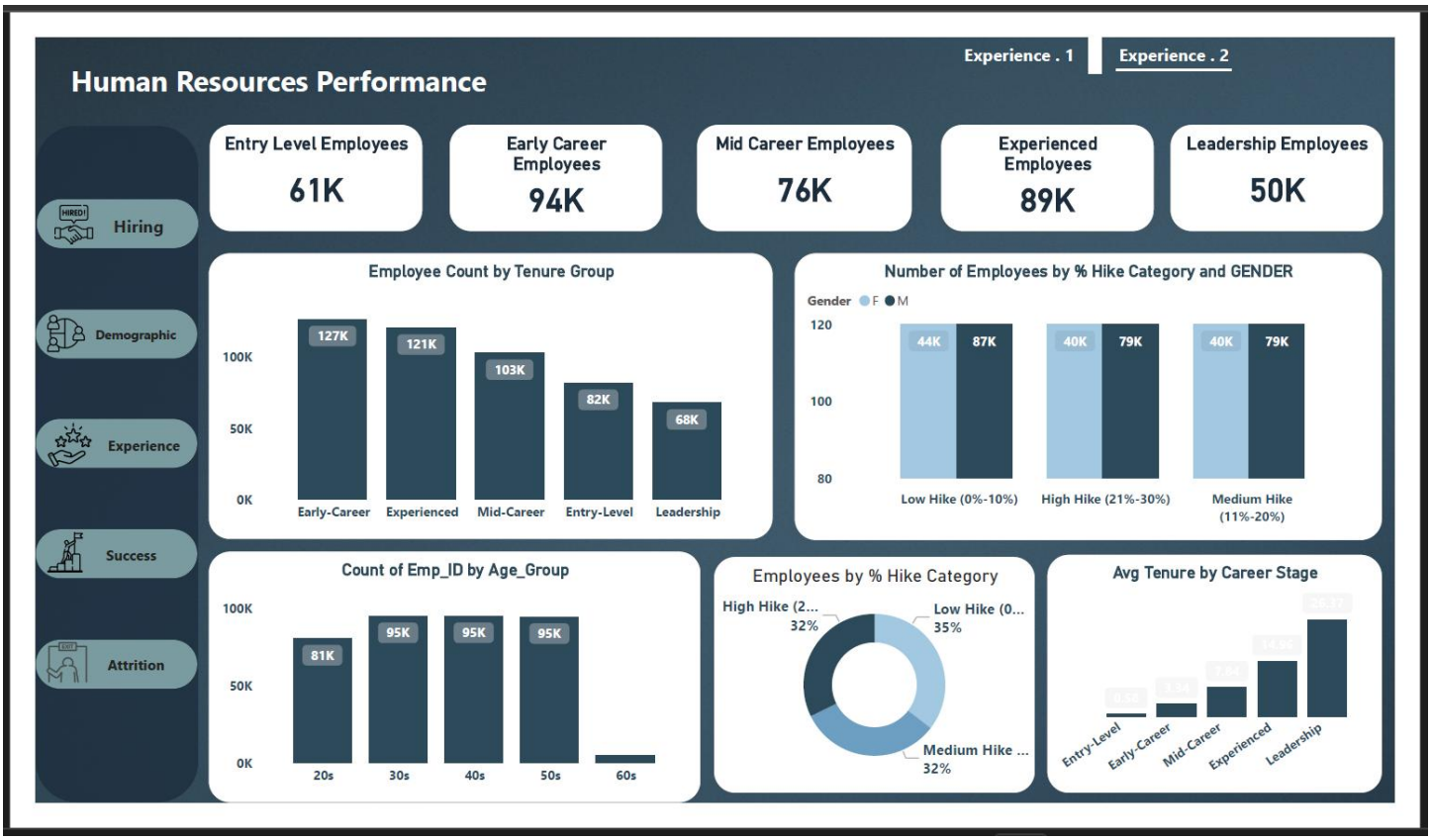


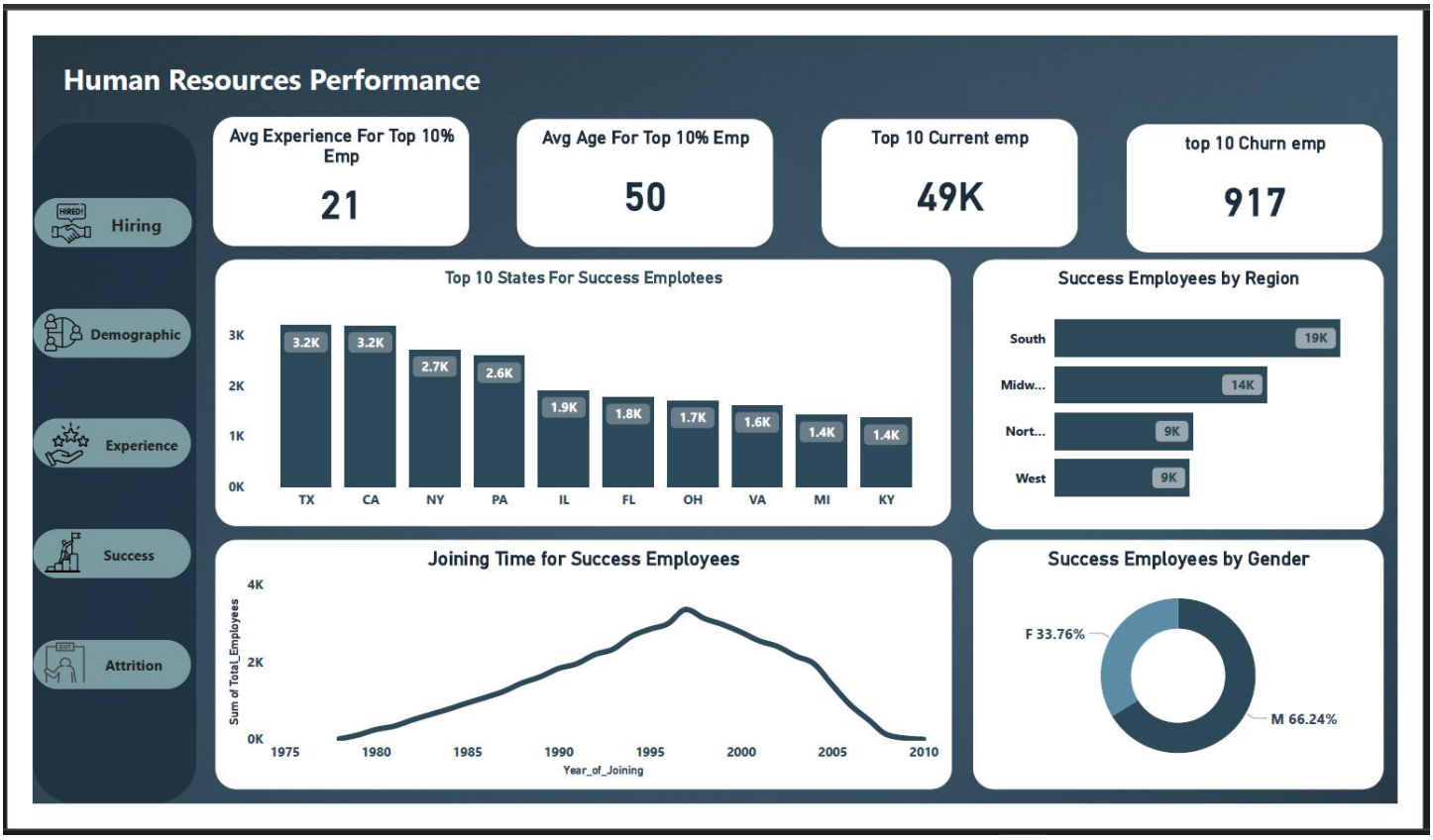


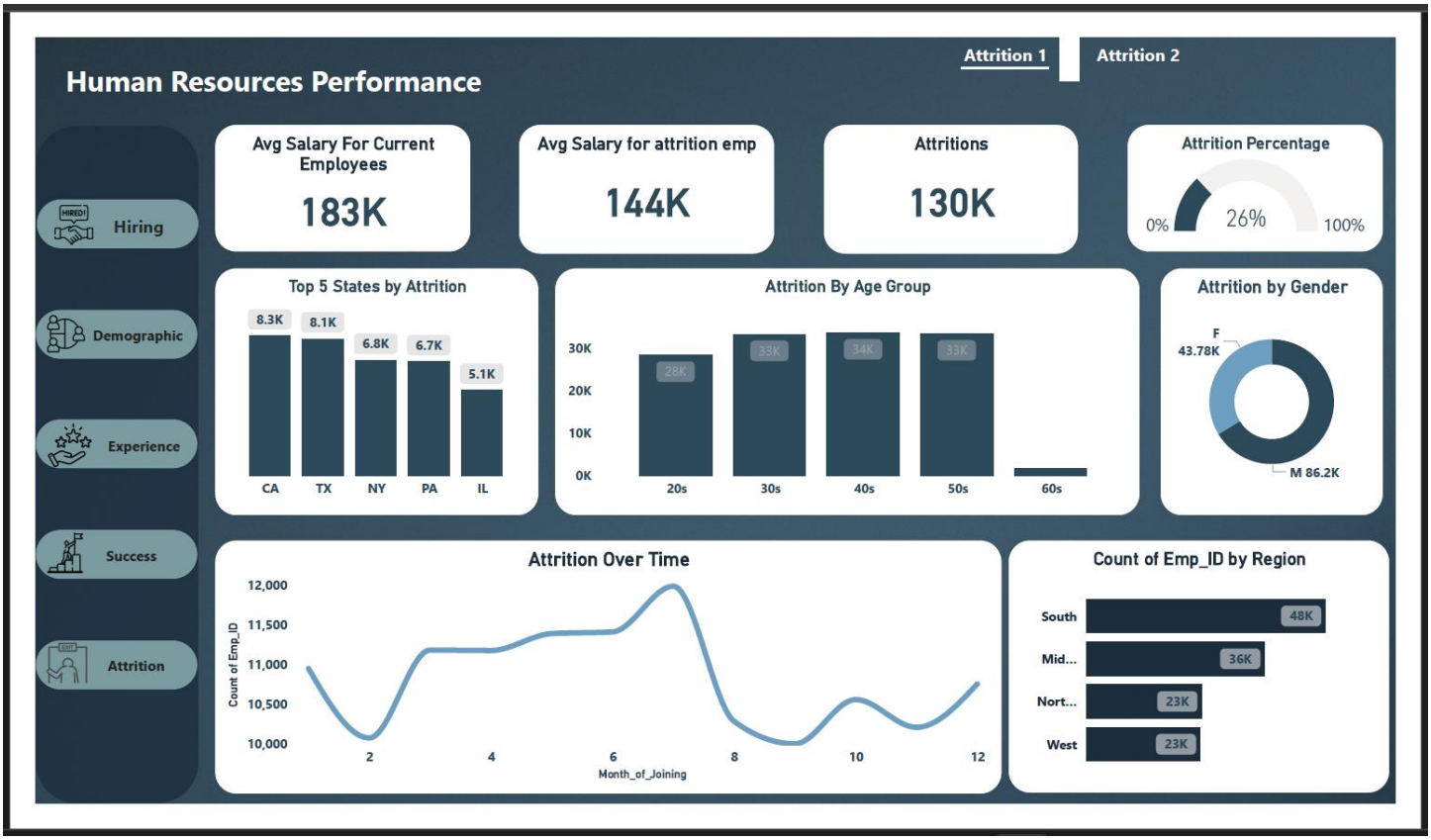

Human Resources Performance
Demographics . 1
Demographics . 2
Hiring
Demographic
Experience
Success
Attrition
Current Males Emp
246K
Current Female Emp
124K
Avg Age At Joining
31
Total States
51
Employees Number by Age Group
109K
128K
129K
128K
6K
100K
50K
0K
20s
30s
40s
50s
60s
Email Type
Individual
53.8%
Company
46.2%
Average Age by Region
Midwest
Northeast
South
West
41
Median of Age_in_Yrs by State
AK
AL
AR
AZ
CA
CO
CT
DC
DE
41

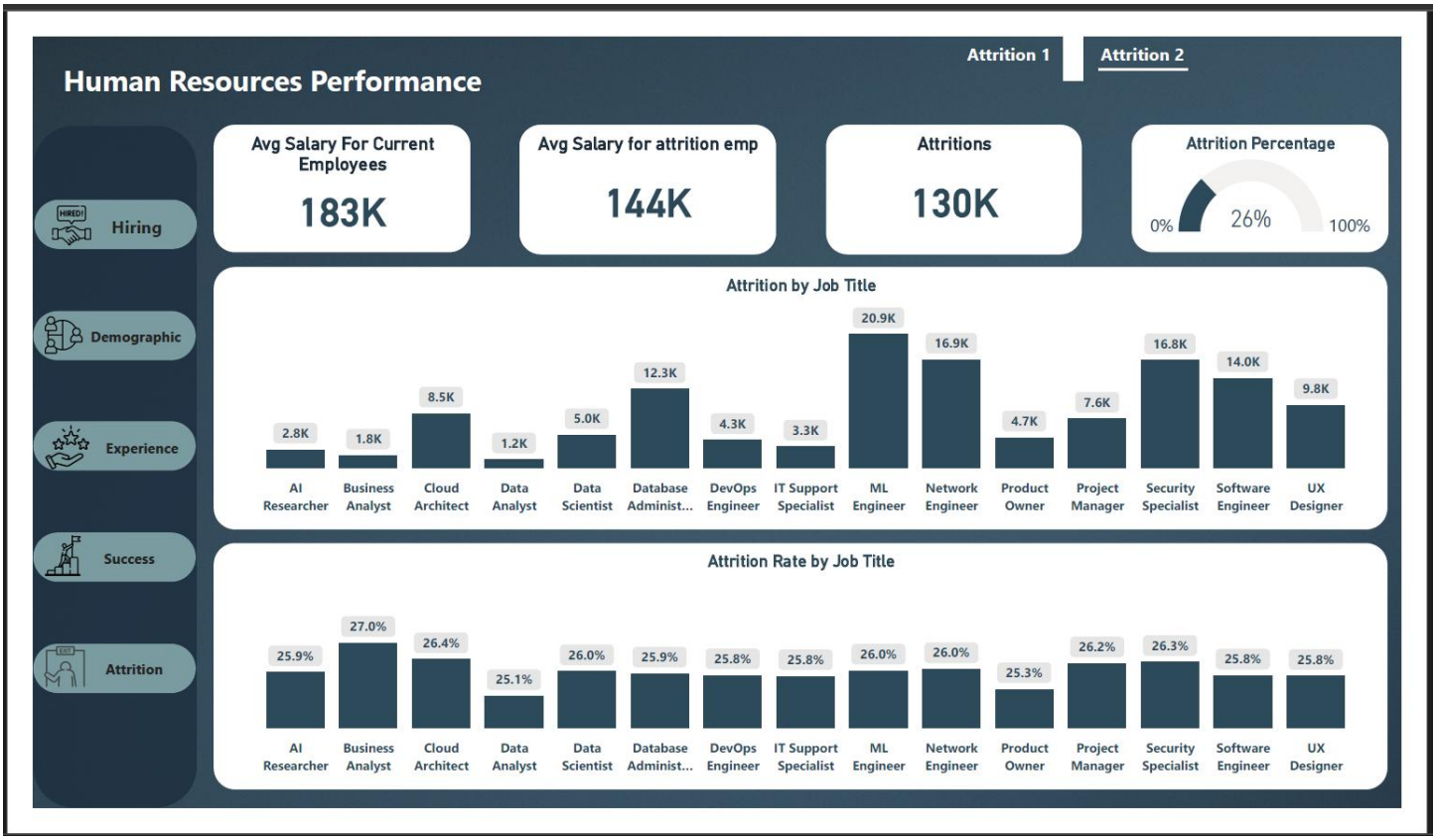

Human Resources Performance
Demographics . 1
Demographics . 2
Hiring
Demographic
Experience
Success
Attrition
Current Males Emp
246K
Current Female Emp
124K
Avg Age At Joining
31
Total States
51
Employees Count by Name Job Title
38K
54K
64K
29K
18K
65K
81K
13K
17K
48K
19K
5K
32K
7K
11K
50K
0K
UX Designer
Software E...
Security Sp...
Project Ma...
Product Ow...
Network En...
ML Engineer
IT Support ...
DevOps En...
Database A...
Data Scientist
Data Analyst
Cloud Archi...
Business A...
AI Researcher
Average Age by Gender
Gend... M F
50
0
48
48
M
F
Averag Salary by Job Title
164K
178K
155K
171K
173K
144K
204K
151K
191K
157K
210K
146K
197K
161K
185K
0.2M
0.1M
0.0M
UX Designer
Software En...
Security Spe...
Project Man...
Product Ow...
Network En...
ML Engineer
IT Support S...
DevOps Eng...
Database A...
Data Scientist
Data Analyst
Cloud Archi...
Business An...
AI Researcher
Employees Stayed vs Left
Left 26%
Stayed
74%


Human Resources Performance
Experience . 1
Experience . 2
Hiring
Demographic
Experience
Success
Attrition
Entry Level Employees
61K
Early Career Employees
94K
Mid Career Employees
76K
Experienced Employees
89K
Leadership Employees
50K
Top 5 Job Titles by Avg Salary
210K
204K
197K
191K
185K
Data Scientist
ML Engineer
Cloud Architect
DevOps Engineer
AI Researcher
Bottom 5 States by Avg Salary
157K
155K
151K
146K
144K
Database Administ...
Security Specialist
IT Support Specialist
Data Analyst
Network Engineer
Salary - Experience Correlation
0.72
0.0
1.0
Avg Salary by Age Group
60s
50s
40s
30s
20s
203K
200K
189K
157K
139K
Avg Salary by Gender
F 167K
M 167K
Average Salary by Region
North...
Midw...
West
South
173K
173K
173K
173K


Human Resources Performance
Experience . 1
Experience . 2
Hiring
Demographic
Experience
Success
Attrition
Entry Level Employees
61K
Early Career Employees
94K
Mid Career Employees
76K
Experienced Employees
89K
Leadership Employees
50K
Employee Count by Tenure Group
100K
50K
0K
127K
121K
103K
82K
68K
Early-Career
Experienced
Mid-Career
Entry-Level
Leadership
Number of Employees by % Hike Category and GENDER
Gender F M
120
100
80
44K
87K
40K
79K
40K
79K
Low Hike (0%-10%)
High Hike (21%-30%)
Medium Hike (11%-20%)
Count of Emp_ID by Age_Group
100K
50K
0K
81K
95K
95K
95K
20s
30s
40s
50s
60s
Employees by % Hike Category
High Hike (2...
32%
Low Hike (0...
35%
Medium Hike ...
32%
Avg Tenure by Career Stage
Entry-Level
Early-Career
Mid-Career
Experienced
Leadership


Human Resources Performance
Hiring
Demographic
Experience
Success
Attrition
Avg Experience For Top 10% Emp
21
Avg Age For Top 10% Emp
50
Top 10 Current emp
49K
top 10 Churn emp
917
Top 10 States For Success Emplotees
3K
2K
1K
0K
3.2K
3.2K
2.7K
2.6K
1.9K
1.8K
1.7K
1.6K
1.4K
1.4K
TX
CA
NY
PA
IL
FL
OH
VA
MI
KY
Success Employees by Region
South
19K
Midw...
14K
Nort...
9K
West
9K
Joining Time for Success Employees
Sum of Total_Employees
4K
2K
0K
1975
1980
1985
1990
1995
2000
2005
2010
Year_of_Joining
Success Employees by Gender
F 33.76%
M 66.24%


Attrition 1
Attrition 2
Human Resources Performance
Hiring
Demographic
Experience
Success
Attrition
Avg Salary For Current Employees
183K
Avg Salary for attrition emp
144K
Attritions
130K
Attrition Percentage
0%
26%
100%
Top 5 States by Attrition
8.3K
8.1K
6.8K
6.7K
5.1K
CA
TX
NY
PA
IL
Attrition By Age Group
30K
20K
10K
0K
28K
33K
34K
33K
20s
30s
40s
50s
60s
Attrition by Gender
F
43.78K
M 86.2K
Attrition Over Time
Count of Emp_ID
12,000
11,500
11,000
10,500
10,000
2
4
6
8
10
12
Month_of_Joining
Count of Emp_ID by Region
South
48K
Mid...
36K
Nort...
23K
West
23K


Human Resources Performance
Attrition 1
Attrition 2
Hiring
Demographic
Experience
Success
Attrition
Avg Salary For Current Employees
183K
Avg Salary for attrition emp
144K
Attritions
130K
Attrition Percentage
0%
26%
100%
Attrition by Job Title
2.8K
1.8K
8.5K
1.2K
5.0K
12.3K
4.3K
3.3K
20.9K
16.9K
4.7K
7.6K
16.8K
14.0K
9.8K
AI Researcher
Business Analyst
Cloud Architect
Data Analyst
Data Scientist
Database Administ...
DevOps Engineer
IT Support Specialist
ML Engineer
Network Engineer
Product Owner
Project Manager
Security Specialist
Software Engineer
UX Designer
Attrition Rate by Job Title
25.9%
27.0%
26.4%
25.1%
26.0%
25.9%
25.8%
25.8%
26.0%
26.0%
25.3%
26.2%
26.3%
25.8%
25.8%
AI Researcher
Business Analyst
Cloud Architect
Data Analyst
Data Scientist
Database Administ...
DevOps Engineer
IT Support Specialist
ML Engineer
Network Engineer
Product Owner
Project Manager
Security Specialist
Software Engineer
UX Designer