HR

January 15, 2025

0.0.1 Import necessary libraries

[ ]:
```
# Import necessary libraries for data handling, database interaction, and␣
↪display
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from datetime import datetime
from IPython.display import clear_output

# Clear any previous output in the cell
clear_output()
```

0.0.2 Data - HR Data with over 22,000 rows from the year 2000 to 2020.

[ ]:
```
#import csv as a pandas data frame
url = 'https://raw.githubusercontent.com/Ahmed-MOrsy/HR_Dashboard_Portfolio/
↪main/HR.csv'
df = pd.read_csv(url)
df.head()
```

[ ]:
```
           id first_name    last_name  birthdate  gender  \
0  00-0037846      Kimmy   Walczynski   06-04-91    Male
1  00-0041533   Ignatius    Springett  6/29/1984    Male
2  00-0045747     Corbie  Bittlestone  7/29/1989    Male
3  00-0055274       Baxy       Matton  9/14/1982  Female
4  00-0076100    Terrell         Suff   04-11-94  Female

                        race            department  \
0         Hispanic or Latino           Engineering
1                      White  Business Development
2  Black or African American                Sales
3                      White              Services
4          Two or More Races    Product Management

                    jobtitle      location  hire_date  \
```

```
0        Programmer Analyst I  Headquarters  1/20/2002
1            Business Analyst  Headquarters   04-08-19
2  Solutions Engineer Manager  Headquarters   10-12-10
3                Service Tech  Headquarters   04-10-05
4            Business Analyst        Remote  9/29/2010

                   termdate location_city location_state
0                       NaN     Cleveland           Ohio
1                       NaN     Cleveland           Ohio
2                       NaN     Cleveland           Ohio
3                       NaN     Cleveland           Ohio
4  2029-10-29 06:09:38 UTC         Flint       Michigan
```

1 Data Cleaning

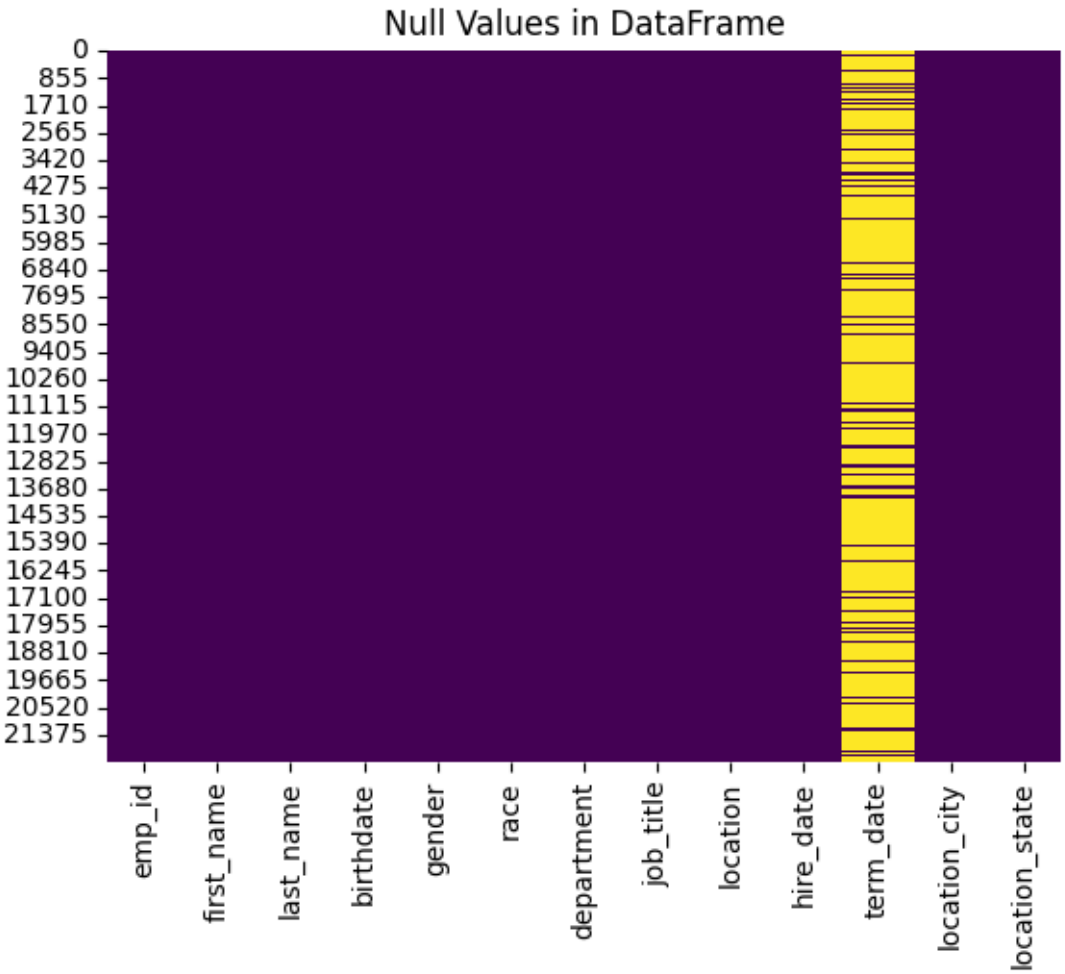
1.1 Rename Columns

[ ]:
```
# Rename the 'id' column to 'emp_id'
df = df.rename(columns={'id': 'emp_id'})
df = df.rename(columns={'jobtitle': 'job_title'})
df = df.rename(columns={'termdate': 'term_date'})
```

[ ]:
```
# Check for null values in each column
null_counts = df.isnull().sum()

# Display the null counts
null_counts
```

[ ]:
```
emp_id               0
first_name           0
last_name            0
birthdate            0
gender               0
race                 0
department           0
job_title            0
location             0
hire_date            0
term_date        18285
location_city        0
location_state       0
dtype: int64
```

[ ]:
```
# Visualize null values using a heatmap
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Null Values in DataFrame')
plt.show()
```



1.2 `normalize_dates` Function

This function standardizes date formats within a Pandas DataFrame column to 'YYYY-MM-DD'.

Purpose:

The `normalize_dates` function addresses inconsistencies in date formats by converting them into a unified 'YYYY-MM-DD' format. This ensures data consistency and facilitates accurate analysis.

Parameters:

- `df` (pd.DataFrame): The input DataFrame containing the date column.
- `column_name` (str): The name of the column containing the dates to be normalized.

Returns:

- `pd.DataFrame`: The DataFrame with normalized date formats in the specified column.

Logic:

1. **Iteration:** The function iterates through each row of the DataFrame using `df.iterrows()`.

2. **Date Conversion:** For each row, it attempts to convert the date value in the specified column to a datetime object using `pd.to_datetime()`. The `errors='coerce'` argument handles invalid dates by setting them to `NaN`.
3. **Formatting:** If the conversion is successful, the date is formatted to 'YYYY-MM-DD' using `strftime('%Y-%m-%d')`.
4. **Update:** The original DataFrame is updated with the normalized date value.
5. **Error Handling:** If a date conversion error occurs, the function prints an informative message and sets the value to `pd.NaT` (Not a Time).
6. **Return:** Finally, the modified DataFrame is returned.

[ ]:
```python
def normalize_dates(df, column_name):
    """
    Normalize date formats in a specified column of a DataFrame to 'YYYY-MM-DD'.

    Parameters:
    df (pd.DataFrame): The input DataFrame.
    column_name (str): The name of the column containing date values.

    Returns:
    pd.DataFrame: DataFrame with normalized date formats.
    """
    for index, row in df.iterrows():
        try:
            # Attempt to convert the date to 'YYYY-MM-DD'
            original_date = row[column_name]
            new_date = pd.to_datetime(original_date, errors='coerce').
 ↪strftime('%Y-%m-%d')
            df.loc[index, column_name] = new_date
        except Exception:
            # Handle invalid dates by setting them to NaN
            print(f"Invalid date format for row {index}: {row[column_name]}")
            df.loc[index, column_name] = pd.NaT  # Assign NaN for invalid dates
    return df
```

[ ]:
```python
# Normalize date formats in a specified column of a DataFrame to 'YYYY-MM-DD'.
normalize_dates(df, 'birthdate')
normalize_dates(df, 'hire_date')
normalize_dates(df, 'term_date')

clear_output()

df
```

[ ]:

	emp_id	first_name	last_name	birthdate	gender	\
0	00-0037846	Kimmy	Walczynski	1991-06-04	Male	
1	00-0041533	Ignatius	Springett	1984-06-29	Male	
2	00-0045747	Corbie	Bittlestone	1989-07-29	Male	

```
3      00-0055274       Baxy       Matton  1982-09-14  Female
4      00-0076100    Terrell         Suff  1994-04-11  Female
…           …        …           …          …     …
22209  99-9797418    Dorella       Garvan  1998-07-08  Female
22210  99-9869877      Dasie      Thorsby  2001-04-19  Female
22211  99-9919822      Nerty      Wilding  2070-02-09  Female
22212  99-9960380    Mabelle        Dawks  1985-09-02    Male
22213  99-9963543    Carroll    Chattaway  1999-07-10  Female

                           race               department  \
0             Hispanic or Latino              Engineering
1                         White     Business Development
2      Black or African American                    Sales
3                         White                 Services
4              Two or More Races       Product Management
…                           …                        …
22209         Hispanic or Latino  Research and Development
22210          Two or More Races                 Services
22211          Two or More Races                 Training
22212          Two or More Races               Accounting
22213                     White              Engineering

                       job_title      location   hire_date   term_date  \
0            Programmer Analyst I  Headquarters  2002-01-20         NaT
1                Business Analyst  Headquarters  2019-04-08         NaT
2      Solutions Engineer Manager  Headquarters  2010-10-12         NaT
3                    Service Tech  Headquarters  2005-04-10         NaT
4                Business Analyst        Remote  2010-09-29  2029-10-29
…                             …           …          …          …
22209        Research Assistant I  Headquarters  2012-02-08         NaT
22210             Service Manager  Headquarters  2017-10-06         NaT
22211              Junior Trainer  Headquarters  2001-02-08         NaT
22212          Staff Accountant I  Headquarters  2005-04-03  2012-12-10
22213       Software Engineer III        Remote  2018-03-27         NaT

      location_city location_state
0         Cleveland           Ohio
1         Cleveland           Ohio
2         Cleveland           Ohio
3         Cleveland           Ohio
4             Flint       Michigan
…             …             …
22209     Cleveland           Ohio
22210     Cleveland           Ohio
22211     Cleveland           Ohio
22212     Cleveland           Ohio
22213    Fort Wayne        Indiana
```

```
[22214 rows x 13 columns]
```

1.3 `fix_birth_dates` Function

The dataset has potential data entry errors in birth dates where the year might be incorrectly recorded with '20' instead of '19' for individuals with a negative value for age.

1.3.1 The Solution:

The function identifies and corrects these errors: 1. **Identification**: It flags rows where the employee's age is negative and their birth year starts with '20'. 2. **Correction**: For flagged rows, it replaces the '20' at the beginning of the birth year with '19'. 3. **Update**: The DataFrame is updated with the corrected birth dates.

1.3.2 Implementation:

The function uses Pandas DataFrame operations and datetime functions to perform the correction, including error handling for invalid dates.

1.3.3 Impact:

This improves data quality, leading to more reliable age calculations and insights from analysis.

1.3.4 Conclusion:

This approach helps maintain data integrity and improves the reliability of analyses based on the HR dataset, ensuring accurate information for decision-making.

[ ]:
```
# Calculate the 'age' column based on the 'birthdate' column
df['age'] = pd.to_datetime('today').year - pd.to_datetime(df['birthdate']).dt.
 ↪year

# Print min and max age
print(f"Minimum age is {df['age'].min()}\nMaximum age is {df['age'].max()}")
```

```
Minimum age is -49
Maximum age is 60
```

[ ]:
```
def fix_birth_dates(df, birthdate_col='birthdate', age_col='age'):
    """
    Fix incorrect birth dates by replacing '20' with '19' in years for people␣
 ↪with negative age

    Parameters:
    -----------
    df : pandas.DataFrame
        Input dataframe containing birth dates and ages
    birthdate_col : str, default='birthdate'
```

```
        Name of the column containing birth dates
    age_col : str, default='age'
        Name of the column containing ages

    Returns:
    --------
    pandas.DataFrame
        DataFrame with corrected birth dates

    Example:
    --------
    >>> df = pd.DataFrame({
    ...     'birthdate': ['2073-03-03', '1995-01-01'],
    ...     'age': [15, 29]
    ... })
    >>> fixed_df = fix_birth_dates(df)
    """
    # Create a copy to avoid modifying the original dataframe
    df_copy = df.copy()

    # Ensure birthdate column is in datetime format
    df_copy[birthdate_col] = pd.to_datetime(df_copy[birthdate_col],␣
↪errors='coerce')

    # Create mask for rows where age is negative and year starts with '20'
    mask = (df_copy[age_col] < 0) & \
           (df_copy[birthdate_col].dt.strftime('%Y').str.startswith('20',␣
↪na=False))

    # Function to correct the year
    def correct_year(date):
        if pd.isna(date):
            return date
        year_str = date.strftime('%Y')
        if year_str.startswith('20'):
            new_year = '19' + year_str[2:]
            return date.replace(year=int(new_year))
        return date

    # Apply the correction only to rows matching the mask
    df_copy.loc[mask, birthdate_col] = \
        df_copy.loc[mask, birthdate_col].apply(correct_year)

    return df_copy
```

[ ]:
```
df = fix_birth_dates(df)
```

```
# Recalculate the 'age' column based on the fixed 'birthdate' column
df['age'] = pd.to_datetime('today').year - pd.to_datetime(df['birthdate']).dt.
 ↪year
```

1.4 Check Term_date in future

[ ]:
```
df['term_date'] = pd.to_datetime(df['term_date'], errors='coerce')

df[df['term_date'] > pd.to_datetime('today')]
```

[ ]:

	emp_id	first_name	last_name	birthdate	gender	\
4	00-0076100	Terrell	Suff	1994-04-11	Female	
27	00-1268049	Fay	Monnelly	1966-07-09	Male	
57	00-2623755	Chrysa	Brownell	1983-04-25	Male	
139	00-6479395	Aura	Steagall	1978-07-19	Male	
173	00-8270076	Raphaela	Clowney	1971-11-02	Male	
…	…	…	…	…	…	
22038	99-1005402	Cornela	Livermore	1969-10-14	Female	
22048	99-1707394	Patrick	Musicka	1976-04-23	Female	
22083	99-3706255	Nappy	Burchess	1999-10-22	Male	
22095	99-4396036	Flory	Hardy-Piggin	1989-03-28	Male	
22127	99-5871990	Brenda	Wank	1983-08-13	Male	

	race	department	\
4	Two or More Races	Product Management	
27	Native Hawaiian or Other Pacific Islander	Engineering	
57	White	Engineering	
139	White	Accounting	
173	White	Engineering	
…	…	…	
22038	Two or More Races	Engineering	
22048	Two or More Races	Human Resources	
22083	Native Hawaiian or Other Pacific Islander	Accounting	
22095	Black or African American	Accounting	
22127	Asian	Training	

	job_title	location	hire_date	term_date	\
4	Business Analyst	Remote	2010-09-29	2029-10-29	
27	Software Engineer I	Headquarters	2010-02-24	2030-03-21	
57	Administrative Officer	Headquarters	2018-02-22	2027-02-01	
139	Staff Accountant I	Headquarters	2013-03-28	2030-02-23	
173	Computer Systems Analyst I	Remote	2010-03-06	2030-08-03	
…	…	…	…	…	
22038	Software Test Engineer I	Remote	2013-06-02	2030-01-30	
22048	Senior Recruiter	Headquarters	2016-11-25	2026-08-01	
22083	Budget/Accounting Analyst II	Headquarters	2017-12-31	2035-08-20	
22095	Administrative Officer	Remote	2019-03-26	2027-03-03	

```
22127    Administrative Assistant I  Headquarters  2012-07-18 2028-03-02

      location_city location_state  age
4             Flint       Michigan   31
27        Cleveland           Ohio   59
57        Cleveland           Ohio   42
139       Cleveland           Ohio   47
173          Warren           Ohio   54
…              …             …  …
22038        Peoria       Illinois   56
22048     Cleveland           Ohio   49
22083     Cleveland           Ohio   26
22095    Fort Wayne        Indiana   36
22127     Cleveland           Ohio   42

[1291 rows x 14 columns]
```

2 Questions

1. What is the gender breakdown of employees in the company?
2. What is the race/ethnicity breakdown of employees in the company?
3. What is the age distribution of employees in the company?
4. How many employees work at headquarters versus remote locations?
5. What is the average length of employment for employees who have been terminated?
6. How does the gender distribution vary across departments and job titles?
7. What is the distribution of job titles across the company?
8. Which department has the highest turnover rate?
9. What is the distribution of employees across locations by city and state?
10. How has the company's employee count changed over time based on hire and term dates?
11. What is the tenure distribution for each department?

[ ]:
```
# Check data types of data
df.dtypes
```

[ ]:
```
emp_id                    object
first_name                object
last_name                 object
birthdate         datetime64[ns]
gender                    object
race                      object
department                object
job_title                 object
location                  object
hire_date                 object
term_date         datetime64[ns]
location_city             object
location_state            object
```

```
age                          int32
dtype: object
```

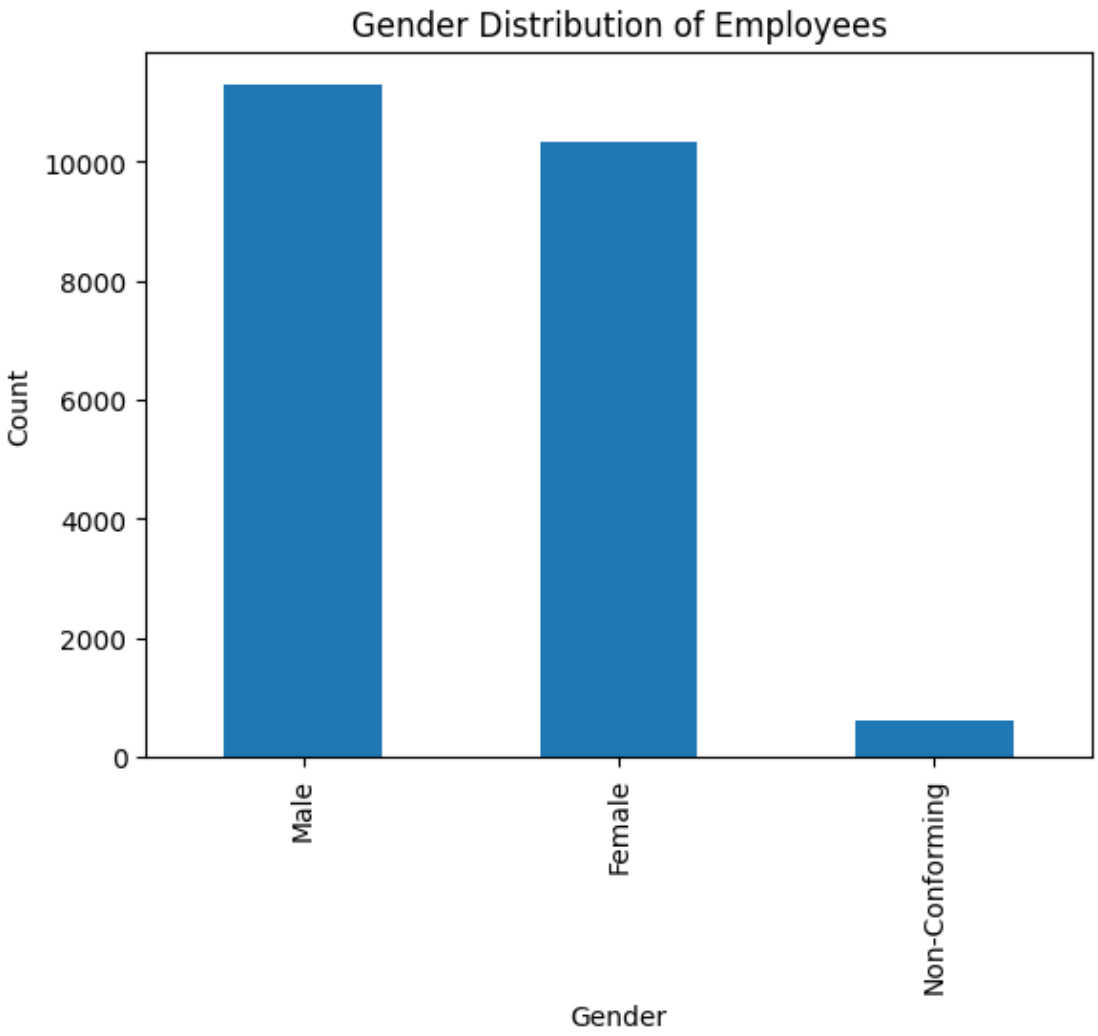
[ ]:
```
# Fix hire_date data type to datetime
df['hire_date'] = pd.to_datetime(df['hire_date'], errors='coerce')
```

2.0.1 1. What is the gender breakdown of employees in the company?

[ ]:
```
# Gender breakdown of employees in the company

gender_counts = df['gender'].value_counts()
gender_counts
```

[ ]:
```
gender
Male               11288
Female             10321
Non-Conforming       605
Name: count, dtype: int64
```

[ ]:
```
# Bar chart of employee gender distribution
gender_counts = df['gender'].value_counts()
gender_counts.plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution of Employees')
plt.show()
```



2.0.2 2. What is the race/ethnicity breakdown of employees in the company?

[ ]:
```
# Race/ethnicity breakdown of employees in the company

race_ethnicity_counts = df['race'].value_counts()
race_ethnicity_counts
```

[ ]:
```
race
White                                    6328
Two or More Races                        3648
Black or African American                3619
Asian                                    3562
Hispanic or Latino                       2501
American Indian or Alaska Native         1327
```

```
Native Hawaiian or Other Pacific Islander    1229
Name: count, dtype: int64
```

2.0.3 3. What is the age distribution of employees in the company?

[ ]:
```
# Calculate the age distribution
age_distribution = df['age'].value_counts().sort_index()

# Age distribution
age_distribution
```

[ ]:
```
age
23    417
24    644
25    632
26    592
27    568
28    604
29    604
30    589
31    589
32    609
33    599
34    583
35    626
36    644
37    647
38    622
39    608
40    636
41    581
42    609
43    593
44    584
45    577
46    590
47    547
48    591
49    614
50    602
51    586
52    597
53    634
54    604
55    557
56    672
57    621
```

```
58    542
59    567
60    133
Name: count, dtype: int64
```

[ ]:
```
# Age group distribution

age_group_counts = df.assign(age_group = (df['age'] // 10) * 10) \
                     .groupby('age_group')['age'].count() \
                     .reset_index(name='count')

age_group_counts
```

[ ]:
```
   age_group  count
0         20   4061
1         30   6116
2         40   5922
3         50   5982
4         60    133
```

[ ]:
```
# Age group gender distribution

age_group_gender_counts = df.assign(age_group = (df['age'] // 10) * 10) \
                            .groupby(['age_group', 'gender'])['age'].count() \
                            .reset_index(name='count')

age_group_gender_counts
```

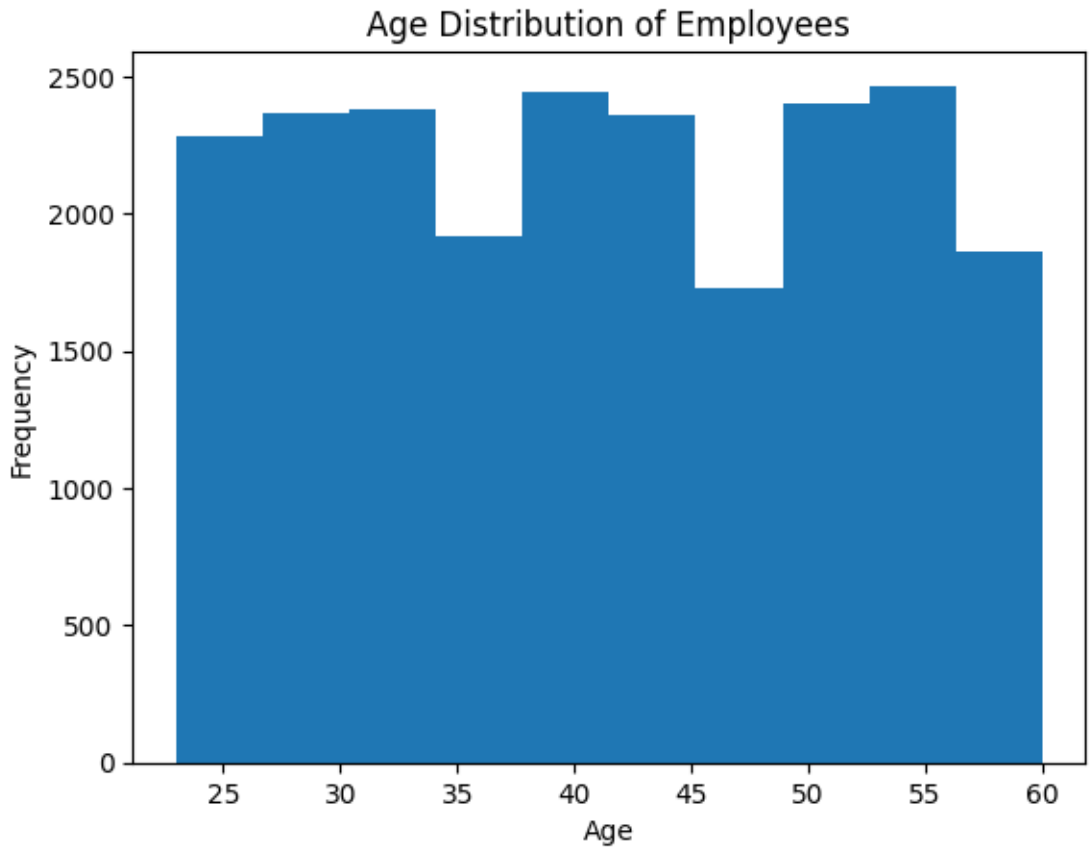
[ ]:
```
    age_group          gender  count
0          20          Female   1851
1          20            Male   2108
2          20  Non-Conforming    102
3          30          Female   2867
4          30            Male   3079
5          30  Non-Conforming    170
6          40          Female   2697
7          40            Male   3065
8          40  Non-Conforming    160
9          50          Female   2853
10         50            Male   2960
11         50  Non-Conforming    169
12         60          Female     53
13         60            Male     76
14         60  Non-Conforming      4
```

[ ]:
```
age_group_gender_pivot = age_group_gender_counts.pivot(index='age_group',␣
↪columns='gender', values='count')
```

```
age_group_gender_pivot
```

[ ]:
```
gender     Female  Male  Non-Conforming
age_group
20           1851  2108             102
30           2867  3079             170
40           2697  3065             160
50           2853  2960             169
60             53    76               4
```

[ ]:
```
# Histogram of employee age distribution
plt.hist(df['age'], bins=10)  # Adjust 'bins' as needed
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution of Employees')
plt.show()
```



2.0.4 4. How many employees work at headquarters versus remote locations?

[ ]:
```
# Employees work on-site vs. remote
location_counts = df['location'].value_counts()
location_counts
```

[ ]:
```
location
Headquarters    16715
Remote           5499
Name: count, dtype: int64
```

2.0.5 5. What is the average length of employment for employees who have been terminated?

[ ]:
```
# Calculate employment length for all employees
df['employment_length'] = (df['term_date'] - df['hire_date']).dt.days

# Filter for terminated employees and calculate average employment length
average_employment_length = df[df['term_date'].notna()]['employment_length'].
 ↪mean()

print(f"The average length of employment for terminated employees is␣
 ↪{average_employment_length:.0f} days")
```

```
The average length of employment for terminated employees is 3814 days
```

2.0.6 6. How does the gender distribution vary across departments?

[ ]:
```
# prompt: How does the gender distribution vary across departments?

# Gender distribution across departments
gender_department_counts = df.groupby(['department', 'gender'])['emp_id'].
 ↪count().unstack()
gender_department_counts
```

[ ]:
```
gender                   Female    Male  Non-Conforming
department
Accounting               1531.0  1711.0            91.0
Auditing                   24.0    28.0             NaN
Business Development      757.0   836.0            49.0
Engineering              3120.0  3373.0           193.0
Human Resources           861.0   904.0            42.0
Legal                     140.0   162.0             9.0
Marketing                 233.0   256.0             5.0
Product Management        277.0   349.0            15.0
Research and Development  513.0   531.0            40.0
Sales                     839.0   946.0            47.0
```

```
Services                  800.0   853.0          33.0
Support                   437.0   481.0          36.0
Training                  789.0   858.0          45.0
```

2.0.7 7. What is the distribution of job titles across the company?

[ ]:
```
# prompt: What is the distribution of job titles across the company?

# Job title distribution across the company
job_title_distribution = df['job_title'].value_counts()
job_title_distribution
```

[ ]:
```
job_title
Research Assistant II             754
Business Analyst                  708
Human Resources Analyst II        613
Research Assistant I              538
Account Executive                 505
                                 ...
Office Assistant II                 1
Associate Professor                 1
VP of Training and Development      1
Office Assistant IV                 1
Assistant Professor                 1
Name: count, Length: 185, dtype: int64
```

2.0.8 8. Which department has the highest turnover rate?

"**Turnover rate**" typically refers to the rate at which employees leave a company or department and need to be replaced. It can be calculated as the number of employees who leave over a given time period divided by the average number of employees in the company or department over that same time period.

[ ]:
```
# prompt: Which department has the highest turnover rate?

# Calculate the turnover rate for each department
turnover_rate = df.groupby('department')['term_date'].count() / df.
 ↪groupby('department')['emp_id'].count()

# Find the department with the highest turnover rate
department_highest_turnover = turnover_rate.idxmax()
highest_turnover_rate = turnover_rate.max()

print(f"The department with the highest turnover rate is␣
 ↪{department_highest_turnover} with a rate of {highest_turnover_rate:.2%}")
```

```
The department with the highest turnover rate is Auditing with a rate of 23.08%
```

[ ]:
```
department_data = df.groupby('department')['emp_id'].agg(total_count='count')
department_data['terminated_count'] = df.groupby('department')['term_date'].
 ↪apply(lambda x: x.notna().sum())
department_data['active_count'] = department_data['total_count'] -␣
 ↪department_data['terminated_count']
department_data['termination_rate'] = department_data['terminated_count'] /␣
 ↪department_data['total_count']
department_data[['total_count', 'terminated_count', 'active_count',␣
 ↪'termination_rate']]
```

[ ]:

department	total_count	terminated_count	active_count	termination_rate
Accounting	3333	586	2747	0.175818
Auditing	52	12	40	0.230769
Business Development	1642	275	1367	0.167479
Engineering	6686	1185	5501	0.177236
Human Resources	1807	309	1498	0.171002
Legal	311	63	248	0.202572
Marketing	494	72	422	0.145749
Product Management	641	114	527	0.177847
Research and Development	1084	212	872	0.195572
Sales	1832	335	1497	0.182860
Services	1686	293	1393	0.173784
Support	954	182	772	0.190776
Training	1692	291	1401	0.171986

2.0.9 9. What is the distribution of employees across locations by state?

[ ]:
```
# Employee distribution across locations by state
location_by_state = df.groupby('location_state')['emp_id'].count()
location_by_state
```

[ ]:
```
location_state
Illinois          868
Indiana           700
Kentucky          451
Michigan          673
Ohio            18025
Pennsylvania     1115
Wisconsin         382
Name: emp_id, dtype: int64
```

2.0.10 10. How has the company's employee count changed over time based on hire and term dates?

This groups the employees by the year of their hire date and calculates the total number of hires, terminations, and net change (the difference between hires and terminations) for each year. The results are sorted by year in ascending order.

[ ]:
```
# Assuming your DataFrame is named 'df' and contains the necessary columns

# Convert 'hire_date' and 'term_date' to datetime objects if they aren't already
df['hire_date'] = pd.to_datetime(df['hire_date'])
df['term_date'] = pd.to_datetime(df['term_date'], errors='coerce')  # Handle␣
 ↪potential errors

# Filter for employees aged 18 or older
filtered_df = df[df['age'] >= 18]

# Extract year from 'hire_date'
filtered_df['hire_year'] = filtered_df['hire_date'].dt.year

# Calculate hires, terminations, net change, and net change percentage
result = filtered_df.groupby('hire_year').agg(
    hires=('emp_id', 'count'),  # Count of hires
    terminations=('term_date', lambda x: x[x.notna()].count()),
    # Count of terminations based on conditions
).reset_index()

result['net_change'] = result['hires'] - result['terminations']  # Calculate␣
 ↪net change
result['change_percent'] = round((result['net_change'] / result['hires']) *␣
 ↪100, 2)  # Calculate percentage
```

```
# Display the results sorted by year
result = result.sort_values('hire_year', ascending=True)
result
```

[ ]:

	hire_year	hires	terminations	net_change	change_percent
0	2000	220	31	189	85.91
1	2001	1122	203	919	81.91
2	2002	1067	174	893	83.69
3	2003	1142	203	939	82.22
4	2004	1135	211	924	81.41
5	2005	1097	207	890	81.13
6	2006	1118	221	897	80.23
7	2007	1090	182	908	83.30
8	2008	1108	190	918	82.85
9	2009	1140	201	939	82.37
10	2010	1099	205	894	81.35
11	2011	1101	187	914	83.02
12	2012	1103	202	901	81.69
13	2013	1105	192	913	82.62
14	2014	1053	177	876	83.19
15	2015	1059	191	868	81.96
16	2016	1122	201	921	82.09
17	2017	1091	190	901	82.58
18	2018	1147	178	969	84.48
19	2019	1083	210	873	80.61
20	2020	1012	173	839	82.91

2.0.11 11. What is the tenure distribution for each department?

How long do employees work in each department before they leave or are made to leave?

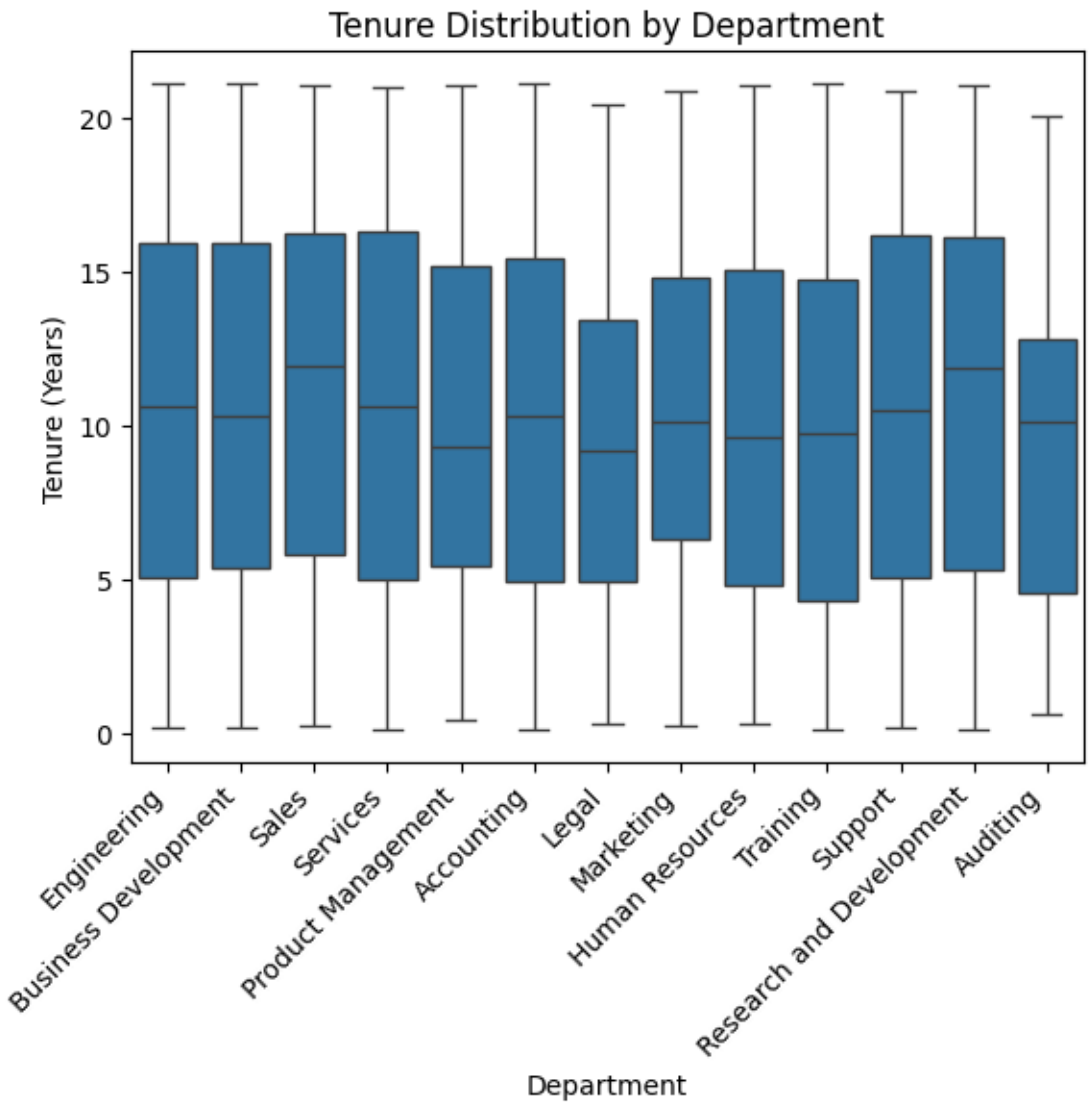
[ ]:
```
# Calculate tenure for each employee in years
df['tenure'] = (df['term_date'] - df['hire_date']).dt.days / 365.25

# Group by department and calculate tenure statistics
tenure_by_department = df.groupby('department')['tenure'].agg(['mean',␣
 ↪'median', 'min', 'max'])

# Sort by mean tenure in descending order
tenure_by_department = tenure_by_department.sort_values('mean', ascending=False)

# Display the tenure distribution for each department
tenure_by_department
```

[ ]:

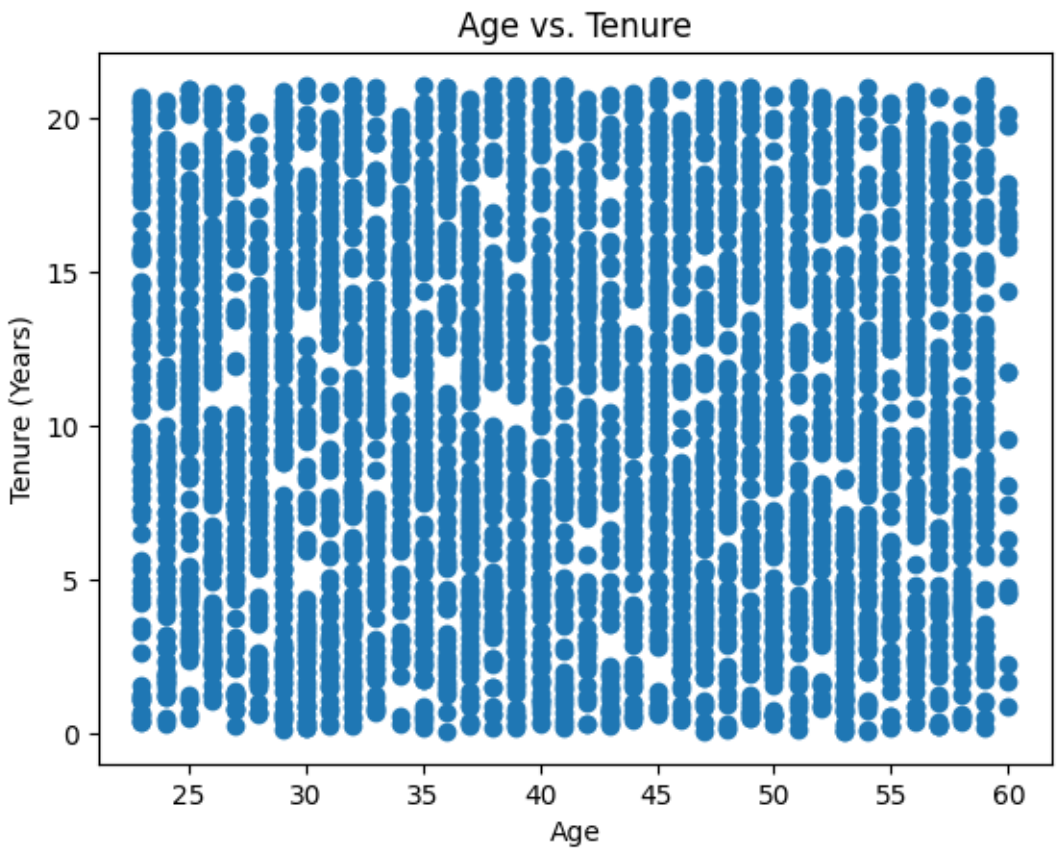
department	mean	median	min	max

```
Sales                       11.114260  11.879535  0.229979  21.056810
Research and Development    10.895058  11.843943  0.114990  21.029432
Services                    10.611162  10.581793  0.106776  20.955510
Engineering                 10.570396  10.565366  0.177960  21.081451
Support                     10.549150  10.468172  0.136893  20.818617
Business Development        10.493595  10.286105  0.134155  21.084189
Marketing                   10.387368  10.080767  0.202601  20.835044
Accounting                  10.347277  10.255989  0.095825  21.084189
Human Resources              9.976409   9.620808  0.287474  21.007529
Product Management           9.940704   9.314168  0.391513  21.037645
Training                     9.682969   9.716632  0.087611  21.070500
Auditing                     9.605749  10.075291  0.607803  20.041068
Legal                        9.355714   9.171800  0.279261  20.388775
```

[ ]:
```python
# Box plot of employee tenure by department
sns.boxplot(x='department', y='tenure', data=df)
plt.xlabel('Department')
plt.ylabel('Tenure (Years)')
plt.title('Tenure Distribution by Department')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels if needed
plt.show()
```



[ ]:
```
# Scatter plot of age vs. tenure
plt.scatter(df['age'], df['tenure'])
plt.xlabel('Age')
plt.ylabel('Tenure (Years)')
plt.title('Age vs. Tenure')
plt.show()
```



---

3 Summary of Findings

1. **Gender Distribution**: There are slightly more male employees (11,288) compared to female employees (10,321). Non-conforming employees represent a very small fraction (605).
2. **Race Distribution**: The majority of employees are White (6,328), followed by Two or More Races (3,648) and Black or African American (3,520). Other races are represented in smaller numbers.
3. **Age Group Distribution**: The largest age group is 35-44 years (6,142 employees), followed by 25-34 years (5,964). The smallest groups are employees aged 55-64 (2,373) and 65+ (0).
4. **Location Distribution**: A large number of employees (16,715) work at the Headquarters compared to Remote locations (5,499).
5. **Average Tenure**: The overall average tenure for employees is approximately 13 years.

[ ]:
```
df.head()
```

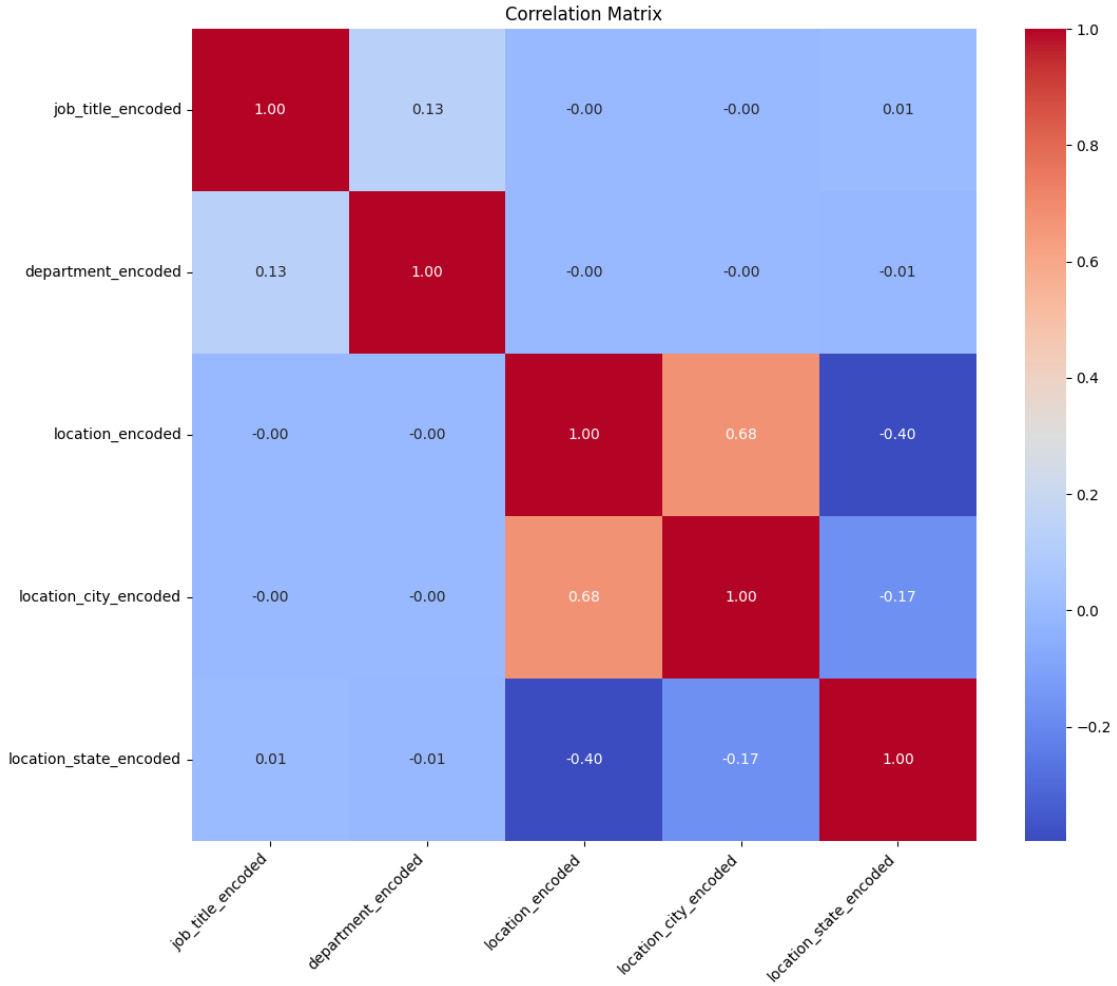
[ ]:
```
       emp_id first_name    last_name  birthdate  gender  \
0  00-0037846      Kimmy   Walczynski 1991-06-04    Male
1  00-0041533   Ignatius    Springett 1984-06-29    Male
```

```
2  00-0045747     Corbie  Bittlestone 1989-07-29    Male
3  00-0055274       Baxy       Matton 1982-09-14  Female
4  00-0076100    Terrell         Suff 1994-04-11  Female

                        race            department  \
0         Hispanic or Latino           Engineering
1                      White  Business Development
2  Black or African American                 Sales
3                      White              Services
4          Two or More Races    Product Management

                    job_title      location  hire_date  term_date  \
0        Programmer Analyst I  Headquarters 2002-01-20        NaT
1            Business Analyst  Headquarters 2019-04-08        NaT
2  Solutions Engineer Manager  Headquarters 2010-10-12        NaT
3                Service Tech  Headquarters 2005-04-10        NaT
4            Business Analyst        Remote 2010-09-29 2029-10-29

  location_city location_state  age  employment_length    tenure
0     Cleveland           Ohio   34                NaN       NaN
1     Cleveland           Ohio   41                NaN       NaN
2     Cleveland           Ohio   36                NaN       NaN
3     Cleveland           Ohio   43                NaN       NaN
4         Flint       Michigan   31             6970.0  19.08282
```

[ ]:
```
# Create a LabelEncoder object
encoder = LabelEncoder()

# Encode the specified categorical columns
for column in ['gender', 'race', 'department', 'job_title', 'location',␣
 ↪'location_city', 'location_state']:
    df[column + '_encoded'] = encoder.fit_transform(df[column])
```

[ ]:
```
# Select the desired columns for the correlation matrix
selected_columns = ['job_title_encoded', 'department_encoded',␣
 ↪'location_encoded',
                    'location_city_encoded', 'location_state_encoded']

# Create a subset of the DataFrame with the selected columns
subset_df = df[selected_columns]

# Calculate the correlation matrix
correlation_matrix = subset_df.corr()

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(12, 10))  # Adjust figure size as needed
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
```

```
plt.title('Correlation Matrix')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels if needed
plt.yticks(rotation=0)  # Keep y-axis labels vertical
plt.show()
```



[ ]:
```
!pip install nbconvert
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!jupyter nbconvert --to pdf HR.ipynb
from google.colab import drive
drive.mount("/content/drive")

# Clear any previous output in the cell
clear_output()
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-
packages (7.16.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
```

```
packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (6.2.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.3.0)
Requirement already satisfied: markupsafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (3.0.2)
Requirement already satisfied: mistune<4,>=2.0.3 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (3.0.2)
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.10.1)
Requirement already satisfied: nbformat>=5.7 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (24.2)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (2.18.0)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (1.4.0)
Requirement already satisfied: traitlets>=5.1 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (5.7.1)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach!=5.0.0->nbconvert) (0.5.1)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.10/dist-packages (from jupyter-core>=4.7->nbconvert)
(4.3.6)
Requirement already satisfied: jupyter-client>=6.1.12 in
/usr/local/lib/python3.10/dist-packages (from nbclient>=0.5.0->nbconvert)
(6.1.12)
Requirement already satisfied: fastjsonschema>=2.15 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=5.7->nbconvert) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=5.7->nbconvert) (4.23.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->nbconvert) (2.6)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (24.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (2024.10.1)
```

```
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.22.3)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (24.0.1)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.10/dist-packages (from jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.3.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.1->jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert) (1.17.0)
```

[ ]:
```
!jupyter nbconvert --to pdf --output HR_Dashboard.pdf /content/drive/MyDrive/
↪Colab\ Notebooks/HR.ipynb
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab
Notebooks/HR.ipynb to pdf
[NbConvertApp] Support files will be in HR_Dashboard_files/
[NbConvertApp] Making directory ./HR_Dashboard_files
[NbConvertApp] Writing 142794 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 366090 bytes to /content/drive/MyDrive/Colab
Notebooks/HR_Dashboard.pdf
```